Music Maker: Using music to introduce coding and concurrency to young learners

Dhananjai M. Rao
Computer Science and Software Engineering Department
Miami University
Oxford, OHIO 45056
Email: raodm@miamiOH.edu

***Abstract*—Computing has become ubiquitous in many fields. The pervasiveness of computing has catalyzed growing demand for a broad range of skills. However, recent trends suggest a growing gap between supply-versus-demand for programmers – estimates indicate that by 2024, there will be a million more jobs than computing professionals resulting in over $500 billion in lost salaries. A key aspect to this growing gap is attributed to negative stereotypes associated with coding, particularly among women. Since stereotypes and perceptions form at a young age, it is imperative to dispel them by exposing young learners to key aspects of coding.**

Accordingly, we have explored introducing coding to elementary school students (4th and 5th grade students in Mid-West USA) in a class involving coding music. The class used custom software called MUSIC MAKER (developed by us) that enables composing music using intuitive notations for notes and meta-notes. MUSIC MAKER enables multitrack composition which is used to introduce basic notions of concurrency to students.

The survey data collected from the students (N = 77) shows that the proposed approach is effective in meeting its primary objective of exciting young learners about coding and initiating (we do not assess long term effects) a more positive outlook on computing as a profession.

***Index Terms*—Elementary school students, Music, Meta notes, Concurrency, Introductory Coding**

I. INTRODUCTION

Computing continues to gain importance and plays a pivotal role in modern society and in many fields, including: medicine, engineering, finance, and defense, to list a few. Consequently, there is a growing demands for programmers and computing professionals with different skill sets. The Bureau of Labor Statistics (BLS) projects that "Computer occupations" will continue to grow at the rate of 12.5% [1]. Furthermore, computational thinking is now considered a fundamental skill for everyone, similar to math or science education [17].

Projected negative impacts: The growing demand for computing professionals to fill high paying jobs has not catalyzed a corresponding growth in computer science graduates. The National Center for Education Statistics [6] estimates that even considering just STEM disciples, only about 8% of STEM degrees are awarded in computer science. Given the estimated job growth rate of 12.5% by BLS [1], the Information Technology and Innovation Foundation (ITIF) estimates that – by 2024 there will be a million more jobs in computing than the number of graduates to fill them resulting in potential loss of about $500 billion in salaries (refer to [11] for details).

Issue: Stereotypes about computer science: Several researchers have investigated a broad range of factors that impede involvement and retention of students in computing [5]. A common theme that has emerged from many investigations is the negative stereotype associated with computer science, particularly among women [2]. Programming and computer science are typically viewed as a boring, isolated, and a male profession [2].

Challenge: Dispelling stereotypes in young learners: The negative stereotypes about programming and computer science forms early in childhood [5]. These stereotypes also influence long-term career paths [2], [14]. For instance, by second grade, girls have already formed stereotypes equating boys with math and associated disciplines [4]. Eccles *et al* [5] use analysis of self- and task perceptions from 865 children (ages 7–10) to show that even first graders have highly differentiated self-beliefs for various activities. They infer that perceptions of competence and value noticeably degrade from 1st to 4th grade. Moreover, early childhood has shown to be a critical period for breaking stereotypes about traditionally masculine STEM fields [14]. The inferences from prior investigations strongly suggest that dispelling stereotypes and reinforcing positive perceptions about programming and computer science in children is important.

A. Proposed method

The aforementioned challenges have motivated creation of several mainstream coding initiatives [3], [10], [13] which are typically targeted to middle school or older students. In many cases, the terms "coding" and "programming" is used interchangeably. In contrast, this paper focuses on introducing coding to elementary school students (ages 9–11) using music composition as the target application. Coding music is introduced as a precursor to programming. Music composition has been chosen because it is universally attractive and can provide instant gratification to children [8], [9], [13]. In order to enable coding music with short learning curve, a custom software called MUSIC MAKER has been developed. As discussed in Section II, MUSIC MAKER also enables multitrack composition, which is used to introduce basic notions of concurrency to students. Data collected from our assessment protocol (see

978-1-5090-5920-1/17/$31.00 ©2017 IEEE

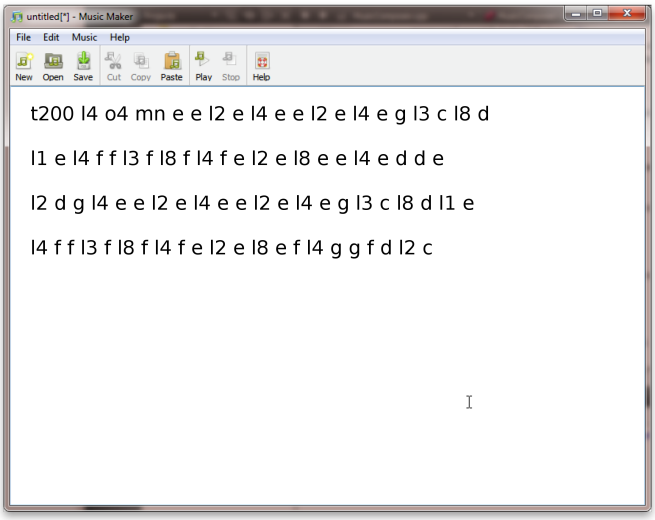


Fig. 1. Screenshot of MUSIC MAKER with code in Music Macro Language (MML) [15]

Section IV and Section V), shows that the proposed method is effective in establishing a positive opinion about coding in 4th and 5th grade students.

II. MUSIC MAKER

Music is universally attractive to all age groups, including elementary school students [9], [13]. Sophisticated programmable hardware and software are available for digital composition of music [8]. Unfortunately, sophistication is accompanied by complexity and complexity is intimidating to young learners. Consequently, to make programming music easier (*i.e.,* very short learning curve), a custom software system called MUSIC MAKER has been developed.

MUSIC MAKER is a highly customized software program that eases composition of instrumental music. The systems has been developed in C++11 using the `Qt-5` (https://www.qt.io/developers/) library (available under GNU Lesser General Public License) to provide a platform independent Graphical User Interface (GUI). Figure 1 shows a screenshot of MUSIC MAKER. The program is compiled to run natively on Windows™ PCs, Mac, and Linux. Using native code enables MUSIC MAKER to be downloaded and run without requiring any additional software installation. Furthermore, `Qt-5` also enables converting the application to operate on mobile devices running Android™ or iOS™ operating systems.

The code for composing music is supplied as input to MUSIC MAKER in the form of notes and meta-notes. The syntax and semantics of the input notes follows the straightforward conventions in the Music Macro Language (MML) [15]. MML uses a declarative programming approach [16] by using abstract syntax to express sequence of operations without describing control flow. Other examples of declarative style of programming include regular expressions, Structured Query Language (SQL), and the C++ standard `algorithm` library. Recollect that declarative programming provides higher-level abstractions focusing on "what to do" (*e.g.,* sort, find, play note etc.) rather than "how to do" (*e.g.,* via variables, loops, methods *etc.*of imperative methods) of programming [15].

TABLE I
META-NOTES SUPPORTED IN MUSIC MAKER

Meta-note	Description
O[1..7]	Selects octave (`O1`: low, `O7`: high)
L[1..64]	Length/duration of notes. `L1` is whole note. `L2` is ½ note, `L4` is ¼ note, *etc.*
P[1..64]	Pause or silence. `P1` is pause for a whole note. `P2` is pause for ½ note, `P4` is pause for ¼ note, *etc.*
I[1..6]	Select an instrument. `I1` selects piano.
*t[1..7]	Select track for subsequent notes
*.	Combines tracks together and starts new set of tracks

The core music notes are represented by corresponding letters `A`–`G`. The meta-notes summarized in Table I are used to modify various characteristics of subsequent notes such as: octave, length, volume etc. The meta-note `P`*n* is used to generate a pause for a fixed duration. The meta-note `I`*n* permits selection of different synthetic instrument. The notes for the instruments are essentially generated using trigonometric functions. For example, a piano-like tone is generated using a simple `sine` function. Other waveforms are generated using square, triangle, or tangent functions. Simplicity of notation is emphasized to ensure coding is straightforward and accessible to young learners or elementary school students (typically 8–11 years).

A. Concurrent programming

The meta-note `t`*n* is an extension we have added to MML. This meta-note enables the use of multiple tracks for composing music. Each track can be coded independently using the full spectrum of notes and meta-notes. However, the independent tracks are combined and played simultaneously. The `*.` meta-note is used to indicate that all the concurrent tracks have been coded and they must be combined at the given point in the code. After the tracks are combined, only one default track is made available. However, as described earlier, additional tracks can be created using the `t`*n* meta-note. Although not currently supported, we anticipate using these features to enable introduction of repetition (of notes/tracks) and procedural abstractions for reusing codes.

Using multiple tracks is analogous to the left and right hand concurrently playing independent notes or chords on a piano. Consequently, coding multiple tracks requires coordinating notes to produce correct sounding music. Coordinating or synchronizing notes on multiple independent tracks provides a simple approach to introduce the basics notions of concurrency to young learners.

III. RELATED RESEARCH

A key objective of MUSIC MAKER is to provide an exciting and intuitive introduction to basics of coding and concurrency to young learners. Coding is performed using Music Macro Language (MML) [15], which uses a simple declarative programming style, similar to those supported by regular expressions and relational algebra [16]. We introduce coding

via MUSIC MAKER as a precursor to more elaborate declarative programming such as those supported by Structured Query Language (SQL), XQuery, C++ algorithms *etc.* [16].

Similar to MUSIC MAKER, a large number of software systems and coding initiatives have been proposed to introduce coding to young learners. Here, we discuss only the closely related efforts that use music as the central theme while referring readers to prior research for a more comprehensive survey of approaches [12].

Some of the more popular initiatives that include music components are Scratch [10] and Alice [3]. They use a drag-n-drop style interface to introduce coding to young learners. Their infrastructures have been extended for more elaborate music programming – Ruthman *et al* [13] discuss the use of Scratch's "musical live coding" capabilities to provide an interdisciplinary introductory course to both music and computer science majors. However, the focus of MUSIC MAKER is to provide a quick and exiting introduction to declarative style of coding which can serve as a precursor to a more elaborate programming paradigm. Freeman [7] *et al* propose introducing computer science principles using `EarSketch`, an integrated curriculum, software toolset, and audio loop library. `EarSketch` provides a multitrack environment that facilitates participants to write code in JavaScript or Python. The audience is primarily secondary and early post secondary students. However, MUSIC MAKER provides a more simple declarative style approach to simplify coding for elementary school students.

Kim *et al* [8] discuss the use of music synthesis as means to motivate curiosity in STEM disciplines. In their weeklong summer program, they use a sophisticated real-time programming environment called "Pure Data" to introduce Digital Signal Processing (DSP) methods to high school students. In contrast, MUSIC MAKER is designed to provide a quick introduction to coding to elementary school students. Kolling [9] proposes a programming environment called "Greenfoot" that combines graphical, interactive output along with text-based Java source to provide a multi-modal learning and programming environment. Unlike MUSIC MAKER, "Greenfoot" is designed to be more general purpose and targets older learners (14 years or older).

IV. ASSESSMENT SETTING & PROTOCOL

The effectiveness of the proposed method, involving MUSIC MAKER and coding music, was explored in conjunction with our "Science Week" in summer of 2015. Science week is a collaborative effort between our institution and local elementary schools to foster interest in STEM disciplines. During science week, students from local elementary schools take classes at our University. Typically, students attend classes or laboratory sessions in physical sciences. However, in summer of 2015, the Computer Science and Software Engineering (CSE) department participated in science week. We invited several sections of 4th and 5th grade students to practice coding music using MUSIC MAKER.

A. Institutional Review Board requirements

Miami University's Institutional Review Board (IRB) approved the assessment setting and protocol after 3 iterations of careful review of all ethical, psychological, and safety aspects as the study involved children. Principals of the schools also reviewed the protocol and their feedback was also used to guide collection of assessments. Although participation in our "Science week" was approved by parents/guardians of students, the IRB required additional written consent to be obtained prior to collecting anonymous feedback. The principals and teachers in the schools facilitated distribution and collection of consent forms. The IRB limited all analysis and publication of data collected from students with parental consent. However, in order to avoid any sense of exclusion or partial treatment, the IRB required that feedback be collected (but not used) from all students with their consent.

B. Assessment procedure

The following procedure has been used to collect anonymous feedback used to assess the effectiveness of using MUSIC MAKER to introduce coding and basic notions of concurrency:

1) Each section of students from the elementary schools met in a large computer lab as shown in Figure 2. Each student was assigned a computer for coding music using MUSIC MAKER. Each computer was running Microsoft® Windows 7™ operating system and had headphones (with sanitary wipes for cleaning before use) for listening to music. MUSIC MAKER was preinstalled on each computer. Internet access was temporarily disabled.
2) Each session had about 20–25 students and lasted 50 minutes in which the assessment was conducted as detailed in the next steps.
3) Feedback forms were printed and distributed to all the students at the start of each session. Forms of students with parental consent were distinguished using color of a small star. Students were verbally instructed on filling-in the forms. The students were reminded that the form is not a quiz or a test and filling-in the form was voluntary. The students were instructed to fill-in just the front page of the form that had preliminary questions to gauge their interests and initial perceptions about computing.
4) Next, the students were introduced to MUSIC MAKER via hands-on presentation. First basic idea of using simple music notes (`A`–`G`) was introduced. Next, using meta-notes to change tempo (*e.g.,* `T4`), octave (*e.g.,* `O4`), and instrument (*e.g.,* `I3`) was introduced.
5) Finally, the use of the track meta-note (*e.g.,* `t2`) and merge tracks meta-note (`*.`) was introduced. Analogy to concurrently using left and right hands to play instruments was used in this context to introduce basic notions of concurrency.
6) After the hands-on presentation (that took about 10-12 minutes), the students were asked to code short music using a 1-page handout. The handout had code for 3 short nursery rhymes. Next students were encouraged to

Fig. 2. Laboratory with a section of elementary school students

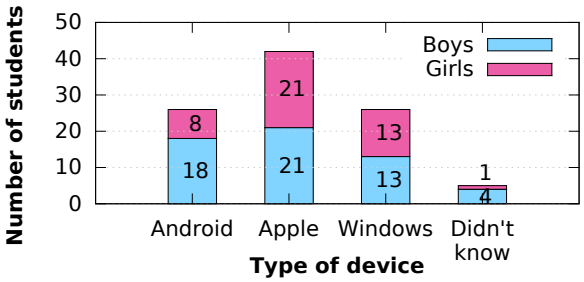
develop and share their own music compositions using MUSIC MAKER.

7) In the last 10 minutes of the session, the students were instructed to complete back (*i.e.,* page 2) of the assessment form. Page 2 had 4 questions repeated from the front to determine if perceptions had changed.
8) At the end of each session, assessment forms were collected from the students and only those with written parental consent were used for further analysis.

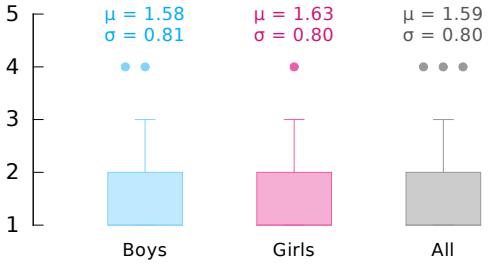
V. ASSESSMENT RESULTS

We conducted classes for 3 sections of 4th graders (65 students) and 3 sections of 5th graders (70 students) on two consecutive days. Out of the 135 students, only 80 students (~55%) had signed parental consent forms and are included in the analysis. The charts in Figure 3 show the type of computing devices, number of devices, and type of usage reported by the students. The feedback form did not distinguish between usages at school versus at home. On average most students reported using multiple devices and no significant difference in number of devices was observed between girls and boys as shown in Figure 3(b). Since most of the homework is handwritten, the predominant usage of computing devices was for research and entertainment as shown in Figure 3(c). Furthermore, the type of use of computing devices was almost consistent between boys and girls. Two of the boys indicated that they had used computers for some coding activities.

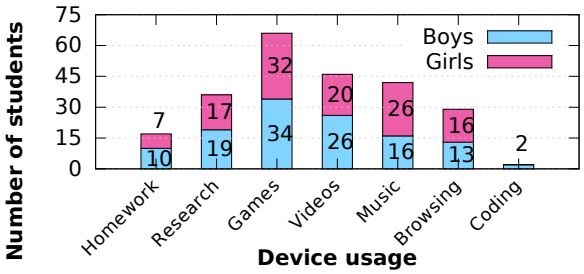
The data in Figure 4 shows statistics on students in the three different categories in the feedback form. About 56% of students indicated that they had heard of coding but did not know how code looks (`Category #1`). Some (~22%) indicated that they had heard of coding and had some idea of how code looks, *i.e.,* `Category #2` in Figure 4. In other words, over 78% of the students had some idea of coding or programming, which suggests they may have formed some opinions about it. On the other hand about 22% of the students had not heard of coding or programming (`Category #3`).



(a) Device types



(b) Devices per student



(c) Type of use

Fig. 3. Types and number of devices along with typical usage reported by students.

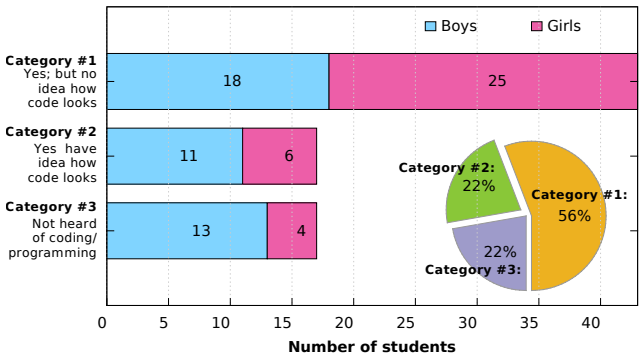
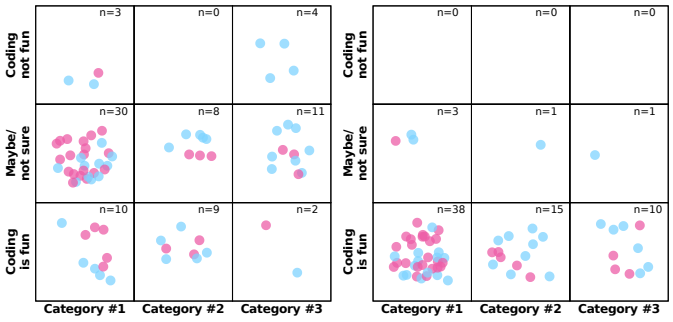


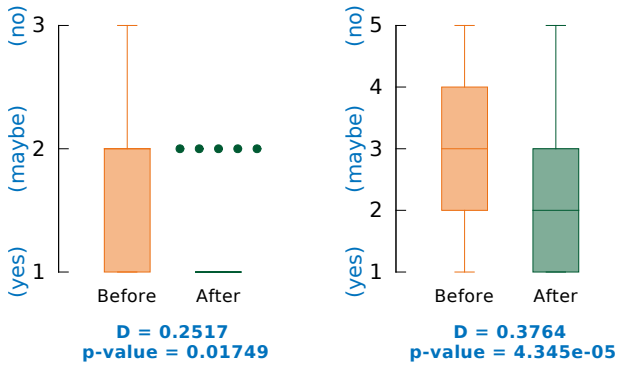
Fig. 4. Data on response to question on familiarity with coding/programming.

The plot in Figure 5(a) shows the initial perceptions of students about coding being a fun activity. Most students had a neutral stance on coding being a fun activity, including about 10% of the students (n=8) who had some ideas on coding. About 25% of the students had a positive impression about coding being a fun activity. The correlogram in Figure 5(b)



(a) Initial perception (b) After using MUSIC MAKER

Fig. 5. Correlation between knowing coding (see Figure 4) and associated perceptions



(a) "Is coding fun?" (b) "Coding as job?"

Fig. 6. Change in opinions "Before" and "After" coding with MUSIC MAKER

shows the impression of students after coding using MUSIC MAKER. As illustrated by the charts in Figure 5, most students (~82%) formed an impression that coding can be a fun activity. In other words, about 66% of the students who were either not sure or had impression that coding is not fun actually changed their opinions. The data also indicated that coding with MUSIC MAKER did not have a negative impact on their opinions – *i.e.,* students did not change their opinions to thinking "coding is not fun".

For further statistical analysis, data associated with two key questions in the feedback form, namely "Do you think coding is fun?" and "Would you like to do coding when you grow up?" were converted by directly mapping responses to a Likert scale. The box plot in Figure 6(a) provides a statistical summary about the change in perceptions after coding with MUSIC MAKER. The `D` metric and `p`-value shown in Figure 6 are statistics from Kolmogorov-Smirnov (KS) two-sample test to show statistical significance of the observed change in students' impressions. Figure 6(a) shows that the change in impression that "coding is fun" is statistically significant with p-value$<$0.05. Analysis of means using a paired t-test also shows that the change is significant with $t(69) = 9.88$, $p \ll 0.001$ being larger than t-critical = 1.9954.

The box plot in Figure 6(b) shows the change in perceptions about potentially pursuing a career in programming. Initially most students were ambivalent but after coding with MUSIC MAKER their opinions changed. As shown in Figure 6(b), the change in opinions about pursuing career in computing is significant (p-value $\ll$ 0.001). Analysis of means using a paired t-test also shows that the change is significant with $t(69) = 5.7$, $p \ll 0.001$ being larger than t-critical = 1.99. The change in response was consistent with observed change in perceptions about programming being a fun activity (*i.e.,* Figure 6(a)). As suggested by the longer tail for the "After" plot in Figure 6(b), two students did not change their mind about pursuing computing and provided written feedback about being interested in other disciplines.

A. Other observations

The students were able to learn coding with MUSIC MAKER quickly. In the first 10 minutes, most students successfully completed their first code from the handouts. The students were also able to easily code their own music using notes and meta-notes. Several students experienced syntax errors from MUSIC MAKER (due to typos in their code) but were able to fix it, with only a few students needing help to identify the issue. The students also seemed to be able to understand and correct semantic aspects of coding – *i.e.,* the music did not sound as they expected and they were able to modify their code appropriately. The case sensitive nature of some of the codes (*e.g.,* `T4` for tempo versus `t4` to select track 4) did not seem to cause confusion for the students. Many students had creative compositions with two or more tracks by the end of the session. This suggests that MUSIC MAKER is effective to introduce basic notions of concurrency to students. Furthermore, students actively shared their compositions with their peers fostering collaboration and interaction between them.

VI. DISCUSSIONS & CONCLUSION

The results from various analyses conducted using feedback from 80 elementary school students are detailed in Section V. Most students estimated that they spent about $3.02_{\pm}1.16$ hours and $2.61_{\pm}1.16$ hours per day using computing devices (total, for various activities in Figure 3(c)) for boys and girls respectively. The time difference between boys and girls was not statistically significant. Coupled with the range of devices used by students (see Figure 3(a)), no significant gender bias was observed in use of computing technologies. Furthermore, both boys and girls reported similar use of technologies for education and entertainment. No gender bias was evident even with gaming, which is often perceived as a male activity.

Statistical analyses did not reveal any strong correlation (for both boys and girls) between initial opinion on "coding is fun" versus: ① type of devices, ② use of computing, or ③ the time spent daily with computers. These observations indicate access and use of technology is not a significant factor influencing initial ambivalence show by the correlogram Figure 5(a) – *i.e.,* students in `category #1` (n=30) who had heard of coding (but did not know how code looks). Consequently, the

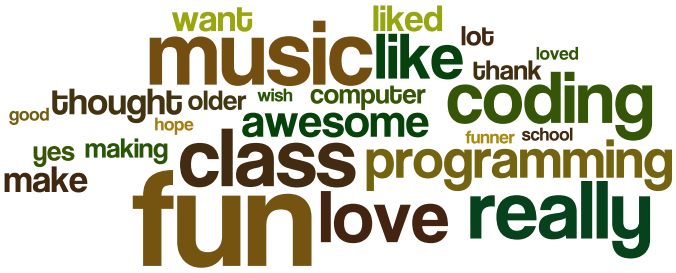


Fig. 7. Top 25 high frequency words in free-form feedback from students

observations suggest other societal factors could play a more dominant role in influencing perceptions.

The charts in Figure 5 clearly show that working with MUSIC MAKER had a positive influence on student perceptions about coding being a fun activity. Even the relatively short 30-minute coding experience of creating music was sufficient to change opinions. The experience suggests that even short classes can have a strong influence, both positive and negative, on young learners. Therefore, considerable care needs to be invested in designing material to introduce coding to students. However, this study did not explore long-term effects, if any, of the proposed approach.

Interestingly, very few students considered that staff music notes could be used a code to program a computer. This suggests, that students do not expect code to use notations from other fields. However, 75% of students identified the input to MUSIC MAKER as being code at the end of the class. Furthermore, all the students found working with the MML notation to be straightforward. Several students managed to develop complex multitrack music. This suggests that the proposed method is effective at introducing the notion of concurrency to students.

The free-form written feedback from students was used to create the tag cloud shown in Figure 7. As suggested by the high frequency words, almost all of the students ($> 95\%$) expressed strong positive sentiment that they had a "fun" experience coding with MUSIC MAKER. They perceived coding to create their own music to be "awesome" and they "really" "liked" / "loved" coding with MUSIC MAKER. Several students explicitly indicated that they liked the idea of concurrency and multitrack coding.

Analysis of feedback collected from the students shows that the proposed approach of introducing coding and concurrency was effective in establishing a positive opinion about coding. The data also shows that MUSIC MAKER is effective in meeting its primary objective of providing an easily accessible (*i.e.,* very short learning curve) but exciting introduction to coding. The universality of music enables reuse of the proposed protocol in other cultural and linguistic settings.

REFERENCES

[1] Bureau of Labor Statistics, "Employment projections: Occupational employment, job openings, and worker characteristics," August 2014. [Online]. Available: https://www.bls.gov/emp/ep_table_107.htm

[2] S. Cheryan, A. Master, and A. N. Meltzoff, "Cultural stereotypes as gatekeepers: increasing girls' interest in computer science and engineering by diversifying stereotypes," *Frontiers in Psychology Roles*, vol. 6, no. 49, pp. 58–71, 2015.

[3] S. Cooper, W. Dann, and R. Pausch, "Alice: A 3-d tool for introductory programming concepts," *J. Comput. Sci. Coll.*, vol. 15, no. 5, pp. 107–116, Apr. 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=364133.364161

[4] D. Cvencek, A. N. Meltzoff, and A. G. Greenwald, "Mathgender stereotypes in elementary school children," *Child Development*, vol. 82, no. 3, pp. 766–779, 2011. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8624.2010.01529.x

[5] J. Eccles, A. Wigfield, R. D. Harold, and P. Blumenfeld, "Age and gender differences in children's self- and task perceptions during elementary school," *Child Development*, vol. 64, no. 3, pp. 830–847, 1993. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8624.1993.tb02946.x

[6] N. C. for Education Statistics, "Webcaspar: Integrated science and engineering resource data system," August 2014. [Online]. Available: https://ncsesdata.nsf.gov/webcaspar/

[7] J. Freeman, B. Magerko, and R. Verdin, "Earsketch: A web-based environment for teaching introductory computer science through music remixing," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '15. New York, NY, USA: ACM, 2015, pp. 5–5.

[8] Y. E. Kim, A. M. Batula, R. Migneco, P. Richardson, B. Dolhansky, D. Grunberg, B. Morton, M. Prockup, E. M. Schmidt, and J. Scott, "Teaching stem concepts through music technology and dsp," in *Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop (DSP/SPE), 2011 IEEE*, Jan 2011, pp. 220–225.

[9] M. Kölling, "The greenfoot programming environment," *Trans. Comput. Educ.*, vol. 10, no. 4, pp. 14:1–14:21, Nov. 2010. [Online]. Available: http://doi.acm.org/10.1145/1868358.1868361

[10] D. J. Malan and H. H. Leitner, "Scratch for budding computer scientists," *SIGCSE Bull.*, vol. 39, no. 1, pp. 223–227, Mar. 2007. [Online]. Available: http://doi.acm.org.proxy.lib.miamioh.edu/10.1145/1227504.1227388

[11] A. Nager and R. D. Atkinson, "The case for improving u.s. computer science education," Tech. Rep., may 2016. [Online]. Available: http://www2.itif.org/2016-computer-science-education.pdf

[12] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson, "A survey of literature on the teaching of introductory programming," *SIGCSE Bull.*, vol. 39, no. 4, pp. 204–223, Dec. 2007.

[13] A. Ruthmann, J. M. Heines, G. R. Greher, P. Laidler, and C. Saulters, II, "Teaching computational thinking through musical live coding in scratch," in *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '10. New York, NY, USA: ACM, 2010, pp. 351–355. [Online]. Available: http://doi.acm.org.proxy.lib.miamioh.edu/10.1145/1734263.1734384

[14] A. Sullivan and M. U. Bers, "Gender differences in kindergarteners' robotics and programming achievement," *International Journal of Technology and Design Education*, vol. 23, no. 3, pp. 691–702, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10798-012-9210-z

[15] Wikipedia, "Music macro language," August 2016. [Online]. Available: https://en.wikipedia.org/wiki/Music_Macro_Language

[16] Wikipedia, "Declarative programming," March 2017. [Online]. Available: https://en.wikipedia.org/wiki/Declarative_programming

[17] A. Yadav, C. Mayfield, N. Zhou, S. Hambrusch, and J. T. Korb, "Computational thinking in elementary and secondary teacher education," *Trans. Comput. Educ.*, vol. 14, no. 1, pp. 5:1–5:16, Mar. 2014. [Online]. Available: http://doi.acm.org.proxy.lib.miamioh.edu/10.1145/2576872