Music Education Meets Computer Science and Engineering Education

Carlos N. Silla Jr.*, André L. Przybysz† and Wellington V. Leal†
*Computer Music Technology Laboratory – Graduate Program in Computer Science (PPGIa)
Pontifical Catholic University of Paraná (PUCPR) – Curitiba, PR, Brazil 80215–901
†Federal University of Technology of Paraná (UTFPR) – Cornélio Procópio, PR, Brazil 86300-000
Email: carlos.sillajr@gmail.com, andrepos@gmail.com, wvidaleal@gmail.com

***Abstract*—This paper presents a novel interdisciplinary approach to aid with the growing concern about how to showcase computer science and engineering degrees to pre-university level students. This novel approach is based on empowering students to create their own music-related game using real music instruments. In order to allow the students to program their games we have used the Greenfoot Introductory Learning Environment to teach object-oriented programming skills with the java programming language. We have also taught the students music theory and practice. All these skills were required to allow the students to successfully develop their musical games that interact with the music instruments.**

I. Introduction

The Brazilian government is one of the Governments worldwide that is concerned with the decreasing enrolling rate and the high drop-out rate in the Engineering degrees. For this reason it has partnered with the Valle S.A. company to promote the thematic call for projects Forma Engenharia. This thematic call had the main objective of supporting Brazilian educators to develop new approaches to attract, engage and showcase different engineering degrees to pre-university students. This particular call was aimed at students who were in up to three years before applying for the Brazilian universities selection processes.

There are different approaches that are used for showcasing (and/or attracting) students to computer science and engineering. The most common approaches are based on using Educational Robots, such as Lego Mindstorms [1], using a circuits-based approaches for electrical engineering awareness [2] or using different softwares for digital game development [1], [3], [4], [5]. Although these approaches successfully showcase and attract students to engineering, they might attract a particular niche of students.

Another area that can be used to showcase students to computer science and engineering is known as Performatics. Performatics has been defined by Ruthmann et al. [6], as a series of courses intended to showcase computer science (CS) by tapping the students inherent interest in performance and the arts. In the context of music performatics [7], students have been taught to build robots that can dance [8] or conduct orchestras [9]. Music concepts have also been used to teach introductory computer science courses [10], [11], [12] or specific topics, such as Design Patterns [13] or Object-Oriented Programming concepts [14]. It has also been used to design new music notation systems [7], to engage students using collaborative music composition and remixing [15], [16] or to design and build electronic music instruments [17].

The main contribution of this paper is to present a novel approach, in the context of music performatics, to showcase students to computer science and engineering that teaches both music and computer science. To the best of the authors knowledge, this is the first project that combines the teaching of music theory, real music instrument playing and computer programming. It also integrates the concept of hardware (using digital music instruments) and software (developed by the students) interaction as the students needed to develop a computer game that interacted with a music instrument.

When developing our novel approach we were faced with important key decisions which we discuss in Section 2. We were also faced with some technical challenges about which music instruments/equipment to use and how (and if) they could be connected to the computers (Section 3). After carefully reflecting on the project format and overcoming the technical challenges we agreed on the final project form that would be used with the students (Section 4). The results achieved in this project by each student are presented in Section 5. Finally the conclusions and future research directions of this project are presented in Section 6.

II. Key Decisions

There is a clear need for more approaches that promote and raise awareness for computer science and engineering. In this section we present the key decisions we had to make to develop our project as they might be helpful to the development of other approaches by other researchers and educators.

A. Which Initial Learning Environment?

The first key decision when developing our approach to attract and/or raise awareness of computer science/engineering was which programming language (if any) should be used. If the approach is focused on developing computer programming skills, it may be the case of employing one of the existing initial learning environments (ILE). Some of the popular choices for ILE's that have been used in related projects are Alice[1]

[1]http://alice.org

[18], Scratch[2] [19], Greenfoot[3] [20] and Processing[4] [21]. In this section we present the lessons learned in the related projects that helped make our decision on which approach to use. There is also a very interesting discussion about the major differences between Alice, Scratch and Greenfoot in [22].

1) Alice: The Alice ILE has been successfully used in many projects. One of these projects is the Dancing Alice [23]. The Dancing Alice project has the goal of attracting more girls to computer science by using Alice actors to dance choreographs designed by the female students. Alice has also been used in the SPIRIT workshop to improve the attitudes and beliefs of high school teachers, counselors, and students toward the field of information technology [24], in the game design approach used by the IT-Adventures outreach program [1] and in Imaginary World Camps outreach program (from 2003 to 2007) to create computer generated movies [3].

In [25] the authors used Alice to teach a Computer Science Introductory (CS0) module. The authors [25] report that while using Alice several at risk students commented to the teaching assistants (TAs) that Computer Science was not as difficult as they thought it would be. However, they also report that when they transitioned their teaching to C++ or java their students became easily frustrated and thought that they did not really have the skills need to pursue a Computer Science degree. [25] conclude that the one of the main reasons for the students frustration was related to the syntax of the languages, because while the Alice ILE allows the students to focus on more important structural programming components and raises the students confidence in programming, this confidence is mostly limited to the ALICE ILE and does not transition to other programming languages.

2) Scratch: The Imaginary World Camps outreach program has switched from Alice to Scratch since 2008 [3]. After making the change in the ILE the students had the option of creating either a music video or a game.

Scratch has been also been used in [6] to teach computational thinking through musical live coding. According to [6] scratch has the ability to generate and play sound using various components. However, in order to make music with more than one sound line in Scratch, it is necessary to address (and teach about) the issue of synchronization. One of the advantages of Scratch, according to [6], is that because it also function as a live interpreter/compiler it could be manipulated in real time for musical live coding.

3) Greenfoot: Greenfoot has been adopted over phyton (that was used in 2007 and 2008) in the game programming course of the COSMOS project in 2009 [4]. According to [4] while using python the students would start witting code for their games without a complete understanding of how it worked. In their opinion [4] it became clear that they should use a language that provides more structure and the safety of static typing and explicit variable declaration.

[2]http://scratch.mit.edu/

[3]http://www.greenfoot.org

[4]http://processing.org

They also noted that Greenfoot introduced some problems as well as many students were confused by the Heavy java syntax such as import statements, class declarations, and the type-casting required for collision detection. Another issue they had with Greenfoot was that it forced them to teach object-oriented programming and the concept of inheritance from the first day of instruction.

4) Processing: Processing was adopted over Greenfoot in the game programming course of the COSMOS project in 2010 [4]. Their main reason for the switch was that Processing, while still being java, offered a rapid visual proto-typing approach.

Processing has also been used in the Musicomputation project [11]. The idea behind this project was to teach teenager musicians about computer science. The reason they chose Processing was because it provides a Java-style syntax, but with libraries geared specifically towards visual art and sound projects. They also provide an interesting discussion about how they feel that Alice or Lego Mindstorms are insufficient to teach the sort of computer literacy that is currently necessary.

B. Which course format?

The second key decision, for developing our novel approach is what format we should use, i.e. how frequently should we meet the pupils, where should we perform the activities related to the project and what type of projects would we work with the pupils. Although the answers to these questions will greatly depend on the goals of the project as well as staff availability, budget, etc. it is noteworthy that some of the related work shows some interesting remarks about these issues that we took into account when developing our approach.

For [26] an on-campus approach to outreach has the advantage of letting students experience what it would be like to be in college pursuing a degree in an engineering discipline and it also allow the utilization of advanced equipment available at the university labs.

Another important aspect of the format is related to the type of projects the students will be engaged with. In the work of [3] where the students had the option of making either a music video or a computer game, the vast majority of students choose to create a game. In [4] the authors report that when the students had the same project, they were not nearly as excited or motivated by the example game they had in previous years. They conclude that creativity and ownership over their work was clearly important to them. For [12] the student projects that had the greatest success were those that were purposeful.

Other reasons for making the students make their own games are provided by [5] who states that ”Digital game-based learning is a popular strategy for engaging students by making learning fun. Actively involving students as designers and producers of digital games may have even greater potential for student empowerment through enhancing concentration and engagement, fostering higher order thinking, and improving learning outcomes”. The use of a computer game construction environment can also be used to encourage female participation in Computer Science [27].

III. Technical Challenges: Which Musical Instruments and Equipment?

Originally, the first decision we were faced with was whether to use acoustic or digital music instruments. In the context of our project we decided to use digital instruments rather than acoustic ones for the following reasons. First, one of our original goals was to develop a project that could be developed in any university or school. Therefore, by using digital instruments it is possible to properly deal with note detections without the need for a sound-proof isolated room where only one student could work at a time. Second, by using digital instruments it is possible to have a computer music technology laboratory with several instruments and computers connected to them, or to just have the instruments stored safely somewhere and just bring them to computer laboratory when necessary. Third, we wanted to have the students working in this project to work in our research laboratory among graduate and undergraduate students, therefore it was mandatory that the students taking part in this project did not disrupt the work of the other students in the lab.

However, even after our decision of using digital music instruments, there were several technical aspects that were not clear. For example, how to connect the digital instruments to the computer? Or how to read the note events generated in the instruments? Did all the instruments use the MIDI standard output in the same way? For this reason, in this section we present the equipment and technical solutions that were useful to the development our project.

A. Keyboard

The keyboard is the most popular digital music instrument; most models come with a MIDI or USB output. In this project we worked with a Roland - Model GW-8 (presented in Figure 1).

Fig. 1: The Roland GW-8 Keyboard used in the project.

B. Digital Drums Kit

In our project we searched for different types of digital drums that would provide a MIDI or a USB output. Our search showed that there are some digital drum kits, from different brands and model that have a USB or MIDI output. However, we chose to use the Digital Drum Kit - Model HD-3 from the Roland manufacturer (presented in Figure 2). In order to connect the HD-3 to the computer, it is necessary to purchase a special cable from Roland, namely the Roland MIDI-UM-ONE cable, to connect the Digital Drums Directly into the PC. It should be noted that although still using MIDI, the drums provide a different set of events to indicate which part(s) of the drum has been hit by the musician. Table I shows the mapping between each piece and the corresponding MIDI code.

Fig. 2: The Roland Digital Drums Kit - Model: HD-3

TABLE I: MIDI Mapping of the parts of the Roland HD-3.

Pad	Note Number	Pad	Note Number
Snare Head	38	Hi-Hat Open Bow	46
Snare Rim	40	Hi-Hat Open Bow	46
Tom 1	48	Hi-Hat Open Edge	26
Tom 2	45	Hi-Hat Close Bow	42
Tom 3	43	Hi-Hat Close Edge	22
Kick Pedal	36	Hi-Hat Foot Close	44
Crash Bow	49	Crash Edge	55
Ride Bow	51	Ride Edge	53

C. Guitar, Bass and the GR-55

The search performed for the digital keyboard and digital drums that could be used in our project were much simpler when compared with the search about how to obtain MIDI or USB output from the Guitar and Bass music instruments. After researching this issue online we discovered that there are two options in order have MIDI generated events for a guitar. The first option is to buy a digital guitar, like the G-5 VG Stratocaster from Roland. The second option is to attach and install specific Roland pickup to any guitar or bass. Due to some limitations in our budget we chose to purchase a standard entry level guitar and a bass and the specific pickups

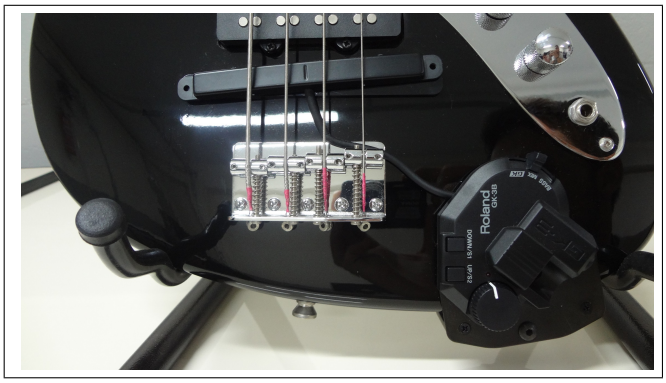


Fig. 3: A standard Bass with the Roland GK-3B Pickup installed.

Fig. 4: Roland GR-55 Guitar Synthesizer.

from Roland. For the Guitar it was necessary to purchase and install a GK-3 Roland pickup. For the bass it was necessary to purchase and install a GK-3B Roland Pickup. Figure 3 shows a picture with a standard bass with the GK-3B pickup installed. It should be noted that only installing the pickups is not enough to connect the Guitar and/or the Bass to the computer.

To connect the guitar and/or bass to the computer, it is necessary to also purchase a Roland Guitar Synthesizer - Model: GR-55 (presented in Figure 4). The setup for using either the guitar or the bass is as follows: First, install the Roland pickup for the guitar/bass. Second, connect the guitar/bass to the GR-55 using a Roland GKC-5 Cable. Third, connect the GR-55 to the PC using a standard USB cable. It should be noted that only one instrument can be connect to the GR-55 at a time and that it is necessary to manually inform the device which instrument has been connected (guitar/bass).

IV. Methodology

The analysis of the related works presented in Section 2 shows that there is not a clear best approach to introduce students to programming, and that many factors must be taken into consideration. Is this project we chose to use the Greenfoot ILE for two reasons. First, since it is java, we would be able to use all existing java packages to deal with the digital music instruments. Second, the students would learn a real programming language. Although, it may be argued that it is not the easiest ILE to start programming with, the fact that it is Java has the benefit that the students can potentially create anything with it.

One key aspect, in the design of the Music Education Meets Computer Science Education, is that we wanted to develop an approach that could be potentially used in any school/university. We also wanted to develop an approach that would tap into the students different interests one subject at a time and that was compatible with their usual school workload. For this reason, our approach is based on having a weekly teaching meeting with the students.

The project is organized into five distinct phases. In the first phase (Weeks 1 to 12), the students are taught how to program in Java using the Greenfoot ILE. In the second phase (Weeks 13 to 24), the students are taught music theory concepts and how to play their particular music instruments. In the third phase (Weeks 21 to 23), the students learn about how to use music processing libraries in Java in order to write programs that can receive events from the digital instruments. In the fourth phase (Weeks 25 to 28), the students have to start thinking about what type of game they will create while learning how to digital photo editing software to create the game interfaces. In the fifth and final phase of the project (Weeks 29 to 36), the students work on their individual projects with the assistance of the project tutors.

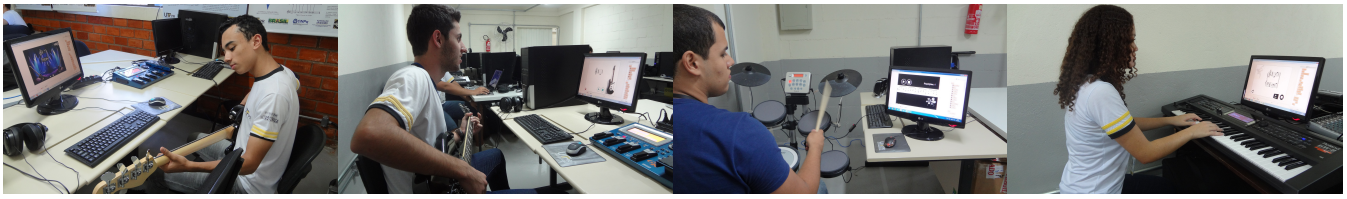
Each student selected to take part in the project had access to his/her own personal computer and respective digital music instrument at the computer music technology laboratory. Although the students had only weekly meetings to learn about different concepts, we always encouraged the students to spend their free time at the laboratory while doing assignments and other course work not related to the project. We felt that this approach would have the added benefit of the students being part of the laboratory.

V. Results

In this section we present the main results of our approach, which are the individual games designed and implemented by each of the students in our project. Figure 5 presents the games developed by each student.

A. The Bass Game

The Bass Game was developed by Gustavo. At the beginning of the project Gustavo picked up the Bass even though he had no prior contact with the instrument. The game he developed has two levels of difficulty. In the first level the player must correctly execute the note being shown in the bass-clef. In this first level the player only sees one note at a time. In the second level of difficulty the player must correctly play a sequence of four notes each time. In the Bass game the player has three lives and every time he makes a mistake he loses one life.



(a) The Bass Game (b) The Guitar game (c) The Drummer game (d) The Keyboard game

Fig. 5: The musical games developed by each student.

B. The Guitar Game

The guitar Game was developed by Alexandre. Alexandre chose the guitar to develop his game, because unlike Gustavo, he already knew how to play the acoustic guitar before participating in the project. However, it should be noted that he had no previous knowledge of music theory and how to play the guitar using a music sheet.

The game developed by Alexandre also had two levels of difficulty. One interesting and notable difference between the guitar game and the Bass Game, is that although they both have two levels of difficulty, the concepts used to develop the levels were different. In the Guitar Game the first level of difficulty shows the player only one note at a time on the tremble Clef (like in the Bass Game). However, it also shows the player where to execute that note in the music instrument. On the second level of the guitar game, the player still had to play one correct note at a time but this time without the visual guide to where to play the note in the guitar.

The main difference between the first level of the bass game and the guitar game is presented in Figure 6. The difference in the design by the two students may be explained by the fact that Alexandre previous learnt how to play the guitar by using a guitar tablature approach. This difference in game design shows that even though the students were using similar music instruments we achieved our objective of letting the students create and have ownership of their work.

C. The Drummer Game

The Drummer Game was developed by Carlos. Carlos already knew how to play the drums before taking part in the project. However, although this was helpful at first, he found some interesting challenges to overcome in comparison with his colleagues. The main reason is that drums sheet music use percussion notation instead of the melodic notation that is used by the other instruments. For this reason, rather that showing which notes should be played it shows the rhythmic pattern that needs to be played. Carlos also implemented an option where the player can set how many times the player needs to play the pattern.

The Drummer Game has three levels of difficulty (Easy, Medium and Hard). In the Easy mode the player can choose between two rhythmic variations commonly used in Pop Music. In the medium difficulty the player can choose between two different rhythm variations one for Rock and another one for Samba, or even practice other drum specific techniques. Figure 7 show an example of the Samba pattern available in the game. In the hard mode the player can choose between another (although harder) rock or Jazz rhythm. The game also has a setup screen where the player chooses how many times he/she needs to repeat a given rhythmic pattern.

D. Keyboard Game

The Keyboard Game was developed by Halanna. At the beginning of the project she chose to work with the Keyboard even though she did not know how to play any musical instrument before. The game she developed has two levels of difficulty. On the first level, the player must correctly play the current note on the music sheet being exhibited on the screen. These notes can be in the tremble clef, as well as in the bass clef.

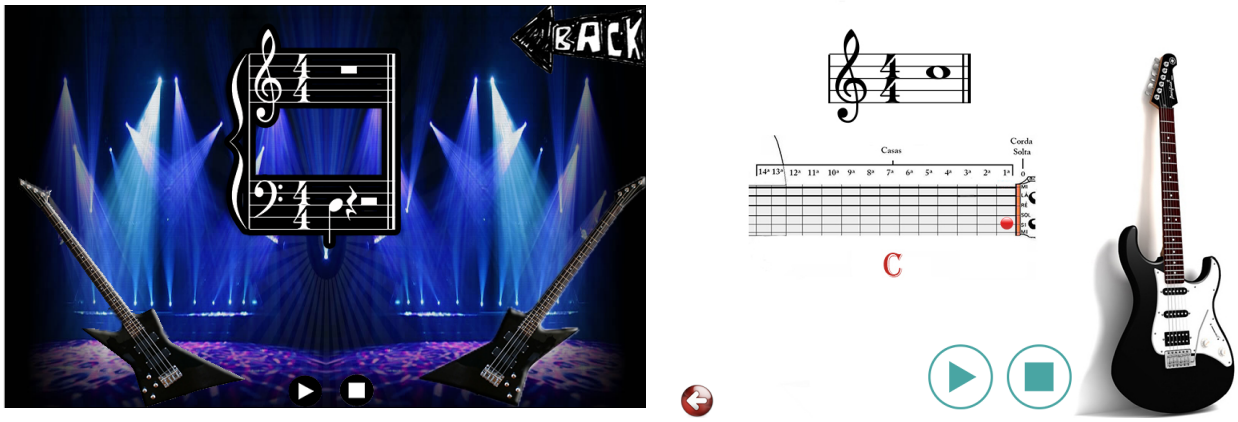
On the second level, the player must correctly play in each turn a sequence of notes. These notes can be all in the same clef or in both of Clefs at the same time. One interesting aspect that Halanna decided to implement in her game was to evaluate the percentage of correct notes played in the sequence and to exhibit a specific screen for different percentages (0%, 25%, 50%, 75% and 100%).

E. Discussion

At the beginning of the project none of the students believed that they would be able to create their own games. However, 36 weeks later, all of them had implemented their very own game, using images they created/edited and that communicated with the digital instrument they choose to work with. The dedication from the students was not the same during the whole project. In the first phase of the project, the students would usually do the minimal amount of work, by going to the weekly classes and solving the programming assignments as quickly as they could.

In the second phase of the project, the students started to become more interested in the project and would often show up between 2 or 3 days at the laboratory. This change in motivation might have been caused by the fact that they wanted to explore and practice their respective digital instruments. This level of interest was kept during the third phase of the project where the students would be curious about how to communicate the events from the musical instruments with the computer.

It was in the fourth and fifth phases of the project, that the students interest grew the most. All students enjoyed working on the creative process of designing their game and



(a) The Bass Game

(b) The Guitar game

Fig. 6: The First Level of the Bass Game (a) and the Guitar Game (b).

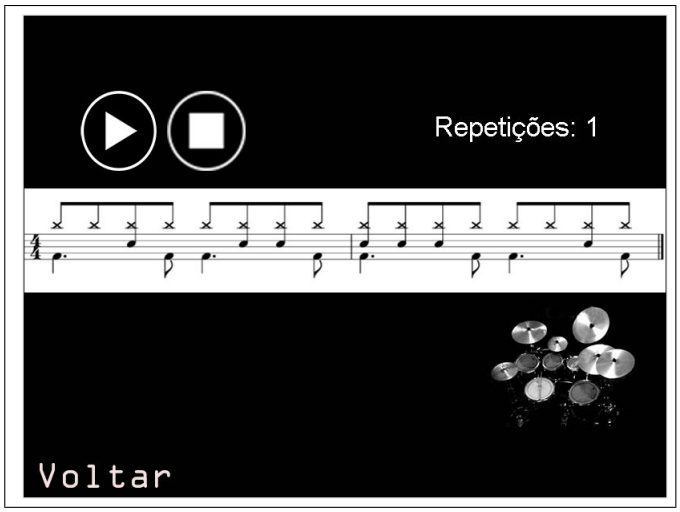


Fig. 7: Example of the Samba rhythmic pattern used in the Drummer Game.

creating/editing the game screens. It should be noted that at first when they were told that they could design anything that they wanted, the students felt a bit lost and we had to intervene by suggesting that they created a game that could help other music students. From this suggestion the students designed the games presented earlier in this section. During the development of their games the students would also be every day in the lab working on their projects. Furthermore, they would often help each other when facing a similar problem that one of them had already solved. This shows that the students also improved their team work skills, even thought they were each working on an individual project.

Based on the growing level of interest in the students with the project, this study corroborates with the findings of previous studies [3], [4], [12], [5], [27]. More precisely, the proposed approach can be seen as a digital game-making approach that empowers students' creativity and individuality. It also benefits from the students interest in music by teaching them music theory and how to play a music instrument (at a beginner level). An added benefit is that our approach integrates the concept of hardware and software integration by using the Digital Music Instruments and the MIDI standard.

VI. Conclusions

In this paper we have presented a novel interdisciplinary approach to showcase students to computer science and engineering. Our approach is inspired by other works from the area of Performatics (i.e. combining computer science and the arts) and more precisely, the area of music performatics (combining music and computer science). To the best of the authors knowledge, our approach is the first approach in the context of music performatics to combine the teaching of both music theory and practice with computer science/engineering.

In order to develop this novel approach we came across two main key design issues: Which (if any) initial learning environment should be used? and which format should be used? After reflecting on the objectives of our project and the lessons learned from previous studies we decided to work with the Greenfoot ILE and to develop a method that could be potentially used in any university or school alongside their usual activities. Given the objectives of our project we were also faced with some technological issues concerning which music instruments and equipment would be necessary to develop our project.

The results of our project with a group of four students showed that the student's interest in the project grew between each of the five different phases of the project. Furthermore, all the students successfully developed their very own musical games that interact with a digital music instrument.

As future research directions we plan to test some variations in the project methodology, e.g. teaching the students music and computer science in parallel. We are also interested in creating novel performatics approaches to showcase computer science and engineering.

ACKNOWLEDGMENTS

The authors thank the financial support from the Brazilian National Council for Scientific and Technological Development (CNPq), the Brazilian Ministry of Science, Technology and Innovation (MCTI), the Valle S.A. company and the Federal University of Technology of Paraná (UTFPR).

REFERENCES

[1] J. A. Rursch, A. Luse, and D. Jacobson, "It-adventures: A program to spark it interest in high school students using inquiry-based learning with cyber defense, game design, and robotics," *IEEE Transactions on Education*, vol. 53, no. 1, pp. 71–79, 2010.

[2] J. Reisslein, G. Ozogul, A. M. Johnson, K. L. Bishop, J. Harvey, and M. Reisslein, "Circuits kit k-12 outreach: Impact of circuit element representation and student gender," *IEEE Transactions on Education*, vol. 56, no. 3, pp. 316–321, 2013.

[3] J. C. Adams and A. R. Webster, "What do students learn about programming from game, music video, and storytelling projects?" in *Proc. of the 43th ACM Technical Symposium on Computer Science Education*, 2012.

[4] G. Smith and A. Sullivan, "The five year evolution of a game programming course," in *Proc. of the 43th ACM Technical Symposium on Computer Science Education*, 2012, pp. 87–92.

[5] Y.-T. C. Yang and C.-H. Chang, "Empowering students through digital game authorship: Enhancing concentration, critical thinking, and academic achievement," *Computers & Education*, vol. 68, pp. 334–344, 2013.

[6] A. Ruthmann, J. M. Heines, G. R. Greher, P. Laidler, and C. Saulters II, "Teaching computational thinking through musical live coding in scratch," in *Proc. of the 41th ACM Technical Symposium on Computer Science Education*, 2010, pp. 351–355.

[7] J. M. Heines, G. R. Greher, and S. Kuhn, "Music performamatics: Interdisciplinary interaction," in *Proc. of the 40th ACM Technical Symposium on Computer Science Education*, 2009, pp. 478–482.

[8] J. Summet, D. Kumar, K. OHara, D. Walker, L. Ni, D. Blank, and T. Balch, "Personalizing cs1 with robots," in *Proc. of the 40th ACM Technical Symposium on Computer Science Education*, 2009, pp. 433–437.

[9] A. Salgian, C. Ault, T. M. Nakra, Y. Wang, and M. K. Stone, "Multidisciplinary computer science through conducting robots," in *Proc. of the 42th ACM Technical Symposium on Computer Science Education*, 2011, pp. 219–224.

[10] A. Misra, D. Blank, and D. Kumar, "A music context for teaching introductory computing," in *Proc. of the 14th ACM SIGCSE Conf. on Innovation and Technology in Computer Science Education*, 2009, pp. 248–252.

[11] A. Meyers, M. C. Cole, E. Korth, and S. Pluta, "Musicomputation: teaching computer science to teenage musicians," in *Proc. of the Seventh ACM conference on Creativity and cognition*, 2009, pp. 29–38.

[12] J. Burg, J. Romney, and E. Schwartz, "Computer science "big ideas" play well in digital sound and music," in *Proc. of the 44th ACM Technical Symposium on Computer Science Education*, 2013, pp. 663–668.

[13] J. Hamer, "An approach to teaching design patterns using musical composition," in *Proc. of the 9th Annual Conf. on Innovation and Technology in Computer Science Education*, 2004, pp. 156–160.

[14] L. Prayaga, "Want to become a music composer?: try with intermediate programming skills," *Journal of Computing Sciences in Colleges*, vol. 27, no. 2, pp. 108–113, 2011.

[15] B. Magerko, J. Freeman, T. McKlin, S. McCoid, T. Jenkins, and E. Livingston, "Tackling engagement in computing with computational music remixing," in *Proc. of the 44th ACM Technical Symposium on Computer Science Education*, 2013.

[16] J. Freeman, B. Magerko, T. McKlin, M. Reilly, J. Permar, C. Summers, and E. Fruchter, "Engaging underrepresented groups in high school introductory computing through computational remixing with earsketch," in *Proc. of the 45th ACM Technical Symposium on Computer Science Education*, 2014, pp. 85–90.

[17] B. Sawyer, T. O. Jason Forsyt and, B. Bortz, T. Finn, L. Baum, I. I. Bukvic, B. Knapp, and D. Webster, "Form, function and performances in a musical instrument makers camp," in *Proc. of the 44th ACM Technical Symposium on Computer Science Education*, 2013, pp. 669–674.

[18] C. W. Herbert, *An Introduction to Programming Using Alice 2.2*, 2nd ed. Cengage Learning, 2010.

[19] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: Programming for all," *Communications of the ACM*, vol. 52, no. 11, pp. 60–67, 2009.

[20] M. Kolling, *Introduction to Programming with Greenfoot: Object-Oriented Programming in Java with Games and Simulations*. Prentice Hall, 2009.

[21] C. Reas and B. Fry, *Processing: A Programming Handbook for Visual Designers and Artists*. MIT Press, 2007.

[22] I. Utting, S. Cooper, M. Kolling, J. Maloney, and M. Resnick, "Alice, greenfoot, and scratch – a discussion," *ACM Transactions on Computing Education*, vol. 10, no. 4, p. Article No. 17, 2010.

[23] S. B. Daily, A. E. Leonard, S. Jorg, S. Babu, and K. Gundersen, "Dancing alice: Exploring embodied pedagogical strategies for learning computational thinking," in *Proc. of the 45th ACM Technical Symposium on Computer Science Education*, 2014, pp. 91–96.

[24] A. Munson, B. M. anb Alka Harriger, T. Lauriski-Karriker, and D. Heersink, "Computing at the high school level: Changing what teachers and students know and believe," *Computers & Education*, vol. 57, no. 2, pp. 1836–1849, 2011.

[25] K. Powers, S. Ecott, and L. M. Hirshfield, "Through the looking glass: teaching cs0 with alice," in *Proc. of the 38th ACM Technical Symposium on Computer Science Education*, 2007, pp. 213–217.

[26] C. E. Davis, M. B. Yeary, and J. J. Sluss Jr., "Reversing the trend of engineering enrollment declines with innovative outreach, recruiting, and retention programs," *IEEE Transactions on Education*, vol. 55, no. 2, pp. 157–163, 2012.

[27] M. Carbonaro, D. Szafron, M. Cutumisu, and J. Schaeffer, "Computer-game construction: A gender-neutral attractor to computing science," *Computers & Educations*, vol. 55, pp. 1098–1111, 2010.