# Using Domain-Specific, Immediate Feedback to Support Students Learning Computer Programming to Make Music

Douglas Lusa Krug
Department of Computer Science
Virginia Commonwealth University
Richmond, VA, USA
Instituto Federal do Paraná - IFPR
União da Vitória, PR, BR
krugdl@vcu.edu

Yifan Zhang
Department of Computer and Information Sciences
University of Delaware
Newark, DE, USA
ericzh@udel.edu

Chrystalla Mouza
College of Education
University of Illinois Urbana-Champaign
Champaign, IL, USA
cmouza@illinois.edu

Taylor Barnett
Department of Music
Virginia Commonwealth University
Richmond, VA, USA
barnettt@vcu.edu

Lori Pollock
Department of Computer and Information Sciences
University of Delaware
Newark, DE, USA
pollock@udel.edu

David C. Shepherd
Department of Computer Science
Virginia Commonwealth University
Richmond, VA, USA
shepherdd@vcu.edu

## ABSTRACT

Broadening participation in computer science has been widely studied, creating many different techniques to attract, motivate, and engage students. A common meta-strategy is to use an outside domain as a hook, using the concepts in that domain to teach computer science. These domains are selected to interest the student, but students often lack a strong background in these domains. Therefore, a strategy designed to increase students' interest, motivation, and engagement could actually create more barriers for students, who now are faced with learning two new topics. To reduce this potential barrier in the domain of music, this paper presents the use of automated, immediate feedback during programming activities at a summer camp that uses music to teach foundational programming concepts. The feedback guides students musically, correcting notes that are out-of-key or rhythmic phrases that are too long or short, allowing students to focus their learning on the computer science concepts. This paper compares the correctness of students that received automated feedback with students that did not, which shows the effectiveness of the feedback. Follow up focus groups with students confirmed this quantitative data, with students claiming that the feedback was not only useful but that the activities would be much more challenging without the feedback.

## CCS CONCEPTS

• **Social and professional topics** → **Informal education**; **K-12 education**; • **Applied computing** → **Sound and music computing**; **Interactive learning environments**.


*ITiCSE 2023, July 8–12, 2023, Turku, Finland*

ACM ISBN 979-8-4007-0138-2/23/07.
https://doi.org/10.1145/3587102.3588851

## KEYWORDS

coding, hint, TunePad, hip-hop

**ACM Reference Format:**
Douglas Lusa Krug, Yifan Zhang, Chrystalla Mouza, Taylor Barnett, Lori Pollock, and David C. Shepherd. 2023. Using Domain-Specific, Immediate Feedback to Support Students Learning Computer Programming to Make Music. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2023), July 8–12, 2023, Turku, Finland.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3587102.3588851

## 1 INTRODUCTION

Attracting students and broadening the participation of underrepresented groups in Computer Science (CS), such as women, African-Americans, and Latino-Americans is a topic that has been a continuous target of CS education research [24]. To attract students of different identities, researchers have experimented with domains outside of those that attract traditional programming students (e.g., robotics and games development) to improve the students' first impression of the field and engage them in activities they enjoy.

One of the non-traditional topics is music, which has shown promise in attracting and engaging students in computer programming [5, 10, 14, 15]. While this approach has shown promise, as with any external domain, there is a major caveat that students must understand two domains at once, computer science and music. To avoid unnecessary barriers to learning, researchers and instructors must pay careful attention to this secondary domain, ensuring that either the students already have the necessary background or that they are given this background. Creating positive experiences (including enjoyment and growing confidence) is particularly important because they profoundly impact students' intention to persist in computing [13].

Learning programming can be challenging, and most students need help to make progress. Therefore, providing timely feedback is an important factor in learning [11] for improving knowledge and acquiring skills [3]. Specifically, immediate automated hints can help students progress in their learning by providing instant and

relevant feedback to correct their mistakes and point them in the right direction to advance through activities [19]. Some studies use feedback with programming activities, which has been shown to be helpful [3, 11]. However, there is still a need for further research on designing programming feedback to create positive, motivating, and engaging programming experiences while promoting performance and learning [17]. Furthermore, the feedback necessary to ensure good musical choices is orthogonal to the feedback necessary to create correct programs; a program can be syntactically correct and even play the correct number of notes but still be woefully out-of-key.

This paper presents the results of a quasi-experimental study that uses music to teach coding in a summer camp format. Students from one group received expert-authored, domain-specific feedback in their in-class activities to help them during the solution process. In comparison, students from another group that received the same instructions had to solve the same activities without integrated feedback. The data collected during eight classes, and twenty-four activities, shows statistically significant evidence that the group that received feedback completed their activities with higher scores. In addition, focus groups conducted after the camp showed that the students valued the feedback and that they increased student confidence.

## 2 RELATED WORK

### 2.1 Teaching Code with Music

Using music to motivate students to learn computer programming has attracted educators' and researchers' attention, resulting in the development of various platforms.

For instance, EarSketch [15], a programming environment for remixing music, uses an approach that focuses on the level of beats, loops, and effects more than individual notes. JythonMusic [16] emphasizes depth, allowing users to generate individual notes or chords instead of mixing existing audio files. Sonic Pi [1], a coding platform initially created to run on a Raspberry Pi, works as a text-based IDE and allows the use of individual notes and also additional samples, such as percussion instruments. TunePad [7] is another approach to engaging students via music mixing, created in a computational notebook style, which integrates visualization and computation, with its focus on usability and ease of use.

The use of music can be effective in teaching introductory computing concepts. Freeman et al. [5], showed that using EarSketch. Also using EarSketch, Magerko et al. [15], showed results that increased the "Computing Confidence", "Motivation to Succeed in Computing" and "Creativity" in students that participated in their workshop. Using TunePad, Horn et al. [10], showed results where students had significant attitudinal gains in interest, self-confidence, enjoyment, and intention to persist in CS. Krug et al. [14], using Sonic Pi, reported a statistically significant difference in engagement towards computer science, with students able to create their own song at the end of the reported summer camp.

### 2.2 Feedback to Improve Code Learning

Formative feedback is defined as information communicated to the learner intended to modify their thinking or behavior to improve learning [25]. An effective feedback is defined as non-evaluative, supportive [25], timely, specific [23, 25], positive, and corrective [23].

The timeliness of feedback can be divided into three categories. First, immediate feedback is often more effective on complex tasks when students have less prior knowledge [25] and is appropriate for novice programmers [17]. Next, delayed feedback is given when the student submits a solution to an auto-grader or test case. This type of feedback is most used to show correct behavior rather than subgoals for a task [17]. Finally, feedback on demand, where the student has to ask for help explicitly, an action that novice programmers need help with [17].

Feedback can also be categorized by type. For instance, "Verification", which informs the learners about the correctness of their responses; "Correct Response", that tells the learner of the correct answer to a specific problem, with no additional information; "Try Again", informs the learner about an incorrect response and allows the learner one or more attempts to answer it; "Error Flagging", which highlights errors in a solution without giving a correct answer; "Elaborated", explains why a specific response was correct or not and may allow the learner to review part of the instruction; "Hints", indicate what to do next, avoiding explicitly presenting the correct answer; And "Bugs/misconceptions", provides information about the learner's specific errors or misconceptions [25].

Successful results in learning, student engagement, and motivation in computer programming using feedback are reported [3, 11]. For instance, Reis et al. [8], report that students using Clara, a tool that provides hints, could significantly reduce their effort to get the correct solution compared to using Python Tutor [9], which produces code visualization, or only test cases. Additionally, students scored Clara as more useful than test cases to fix bugs in their programs [22]. Also, Marwan et al. [17], present an adaptive immediate feedback system integrated into a block-based programming environment. This system provides positive and corrective feedback in real-time as students work. The results show that the feedback system increased students' intention to persist in CS and that students that used the feedback system had greater engagement than the students that did not use the feedback system.

In terms of expert-authored feedback, Gerdes et al. [6], presents *Ask-Elle*, a tutor for learning that supports the stepwise development of Haskell programs by providing hints during the development process. Also using expert-authored feedback, Benotti et al. [2], presents *Mumuki*, a web-based tool that provides formative feedback. One of the differences between *Mumuki* and *Ask-Elle* is that *Mumuki* shows feedback only when the solution is submitted and not during the development of the program.

## 3 BACKGROUND

### 3.1 *Code Beats*

*Code Beats* is an approach that teaches foundational computer programming concepts using music, specifically hip hop. Students taking *Code Beats* classes learn the basics of music theory and coding fundamentals and can apply the concepts in activities that use actual hip-hop songs.

*Code Beats* uses hip-hop as the musical genre for two main reasons: (1) hip-hop is part of the African-American and Latino-American cultures, populations underrepresented in CS, and the

primary focus to broadening participation; (2) a hip-hop beat is created in a way that does not require extensive harmonic knowledge, with its focus on complex rhythms, being well suited to teaching the computational concepts.

We use actual hip-hop songs transcribed to a music-coding platform called TunePad [7]. TunePad is a platform delivered as a website, using a computational notebook approach with a different code snippet for each cell. Each cell is an instrument that is programmable and can be played individually or with all cells at the same time. The code and music from each cell code can be relatively simple, but realistic-sounding songs can be composed by combining many cells. TunePad has a user-friendly, interactive interface where users can experiment and test the instruments by simply clicking on a note from a virtual keyboard or percussion instrument. TunePad uses Python code and pre-built functions to allow the users to create the code in each cell that will form the song.

*Code Beats*' approach is based on the Use-Modify-Create framework [12]. In every class, students work with almost complete projects inspired by an actual hip-hop song containing all but one coding cell (Use). Each project contains one empty cell where the students must create the code according to the project's instructions (Modify). The expected code for this cell is simple and short, and the sound produced by this cell will complement the song from the whole project. By the end of *Code Beats*, the students are expected to create their project, following music and code recommendations (Create).

## 3.2 Domain-Specific Immediate Feedback

In a previous iteration of *Code Beats*, we noticed that even using scaffolded, short tracks, the code developed by many students, while accurate from a computer science perspective, often violated musical guidelines, playing notes out-of-key or in an odd rhythm. While most students could hear, and correct, the most egregious musical mistakes, they often struggled with more subtle issues. This eroded their confidence overall, even when they were mastering the computer programming concepts. With that in mind, we implemented a domain-specific, immediate feedback system to guide students as they completed the activities. The feedback messages point out the music requirements of the activity, specifically where the current composition falls short, and not the coding requirements. In the context of this work, aligned with the feedback types explained by Shute [25], formative feedback will be called "Hints", indicating what to do next, avoiding presenting the correct answer. The other type will be called "Feedback", combining "Verification" and "Error Flagging" from their original types, indicating whether the program is correct or incorrect.

The content of the hints and feedback messages are a breakdown of the activity requirements. When the activity starts, the messages on the screen are hints that inform students of the activity requirements. At the bottom of Figure 1, in yellow, are examples of hints, each pointing to one of the activity's requirements. The first reminds the student that the melody must be four beats long, the second states that the melody must be the same as the original, and the third states that the rhythm must be the same as the original melody. The sound of the original track (e.g., melody or drum track) is provided to students as example.


**Figure 1: Example of Hints**

Each activity has a trigger that transforms the hint into feedback. In our running example, each time the number of beats in a track changes (i.e., the student plays a note or adds a rest), the feedback mechanism is triggered. For example, Figure 2 shows an example of the same activity that is now four beats long. The messages that were hints in Figure 1 (yellow) now are feedback in Figure 2 (green and red). The first and third messages, in green, are examples of verification feedback indicating that the student has achieved two requirements: (1) the melody is four beats long; (2) the rhythm is the same as the rhythm from the original melody. The second message, in red, is an example of verification feedback indicating that the student did not reach that requirement because the melody is not the same as the melody from the original song. The feedback points out that something is wrong and indicates the line of code with the problem, also working as error flagging.


**Figure 2: Example of Feedback - 1**

On the other hand, if the student reaches all the requirements, all messages will be green, using the verification feedback to indicate that the solution is correct. Figure 3 shows an example of the same activity with all feedback messages in green, indicating that the student reached the correct answer.

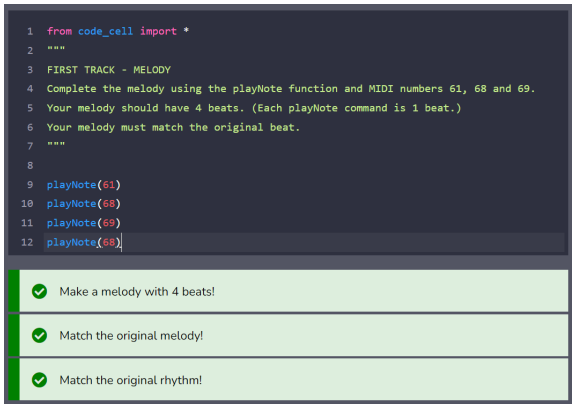


**Figure 3: Example of Feedback - 2**

Observing the classification described by Narciss [21] and extended by Keuning, Jeuring, and Heeren [11], the formative feedback used in the context of this work have the following components: Knowledge of Performance (KP), as we identify the activity subgoals and indicate when they are achieved; Knowledge of Result/Response (KR), identifying the specific subgoal as correct or incorrect; Knowledge of the Correct Results (KCR), identifying when all subgoals are correct, the activity is correct; Knowledge About Task Constraints (KTC), subtype Hints on Task Requirements (TR), breaking down the activity goal in domain-specific hints. Knowledge About Mistakes (KM), subtype Solution Errors (SE), indicating that the solution does not show the behavior expected in the activity, pointing to the line where the error is. Finally, in terms of technique, also defined by Keuning, Jeuring, and Heeren [11], the feedback system reported here uses Basic Static Analysis (BSA), analyzing the piece of code the student is writing in a specific cell to generate the hint or feedback message.

A test case-like code performs the static analysis using as input the cell code and cell output (e.g., Musical Instrument Digital Interface (MIDI) numbers), returning to the activity cell the messages content and its type. This analysis is performed at every line completion in the students' code (e.g., hitting enter, changing line, or "playing" the cell's code). This categorizes the feedback as immediate, as it is there since the beginning of the activities, and it is updated at every code change. The feedback is expert-authored, where the rules are specified according to the expected results of each activity known by the feedback author.

## 4 RESEARCH QUESTIONS

**RQ1**: How does domain-specific, immediate feedback affect the activities' correctness during class?
**RQ2**: What is the student's perception of the domain-specific, immediate feedback?

## 5 METHODS

### 5.1 Study Design and Data

This paper reports a quasi-experiment using data collected from a summer camp held in the Summer of 2022. The classes were in person and held for five days, with two classes each day, each with a duration of one hour and forty-five minutes. Each class consisted of a mix of instructions and coding activities. The coding activities were actual hip-hop songs transcribed into TunePad. The project presented to the student was almost complete, with all song tracks but one to students create (e.g., melody or hi-hat track). The activities were divided into two types. The first was the short activity, which asked students to create a song track in a specific code cell that would mimic the original song. This short activity had only one correct answer, for example, a sequence of musical notes in a particular order and rhythm. And the second was the long activity that asked students to create a song track in a code cell that would fit the original song. This activity had the music requirements but allowed multiple solutions. For instance, they required that the students used a group of musical notes but did not specify the order. Each class, from class one to class eight, had two short and one long activity, summing up to 24 activities [1]. The last two classes did not have these activities, as the classes were used to prepare the students to create their final project.

During the Summer of 2022, two sessions of *Code Beats* were offered, one in the morning and another in the afternoon. The student chose the session that they would like to attend. Students from one session received the activities with the immediate feedback system, and students from the other session received the same activities, except by the immediate feedback system. All the rest of the classes were the same. The group of students that received the immediate feedback will be called "Group A" and the group of students that did not receive the immediate feedback will be called "Group B".

To analyze the correctness of the students' solution, we collected the final solution developed by the student and also code snapshots generated during the solving process. To analyze the immediate feedback system effect, the students from Group A answered the question: "When you saw a message that looked like this, how did it affect your motivation?" presented with each of the hints/feedback examples. Additionally, they participated in a focus group to talk about their experience.

### 5.2 Demographics

Students from both groups answered demographic questions. 24 students from each group answered those questions. On average, Group A students were 12.1 years old (min - 10; max - 15). 33.3% of the students from Group A self-declared as girl, 58.3% as boy, and 8.3% prefer not to say. 12.5% of the students from Group A self-declared as Asian/Asian-Americans, 33.3% as Black/African-American, 8.3% as Multi-Racial, 37.5% as White/Caucasian, and 8.3% prefer not to say. 58.3% of the students from Group A declared that they play instruments, and for 29.2% of the students from Group A, *Code Beats* was their first coding experience.

On average, Group B students were 12.2 years old (min - 10; max - 17). 16.7% of the students from Group B self-declared as girl, 75.0% as boy, and 8.3% prefer not to say. 25.0% of the students from Group B self-declared as Asian/Asian-Americans, 25.0% as Black/African-American, 8.3% as Multi-Racial, 33.3% as White/Caucasian, and 8.3% prefer not to say. 54.2% of the students from Group B declared that they play instruments, and for 16.7% of the students from Group B, *Code Beats* was their first coding experience.

[1]The list of activities can be accessed here: http://bit.ly/3GBpOP1

### 5.3 Data Analysis

To answer the first research question, all activities' final solutions and code snapshots collected during the solving process were analyzed, searching for the code that meets the expected solution. If the code that meets the expected solution was identified at any time, the solution for that student and activity was classified as "Correct", even if the student changed the solution afterward. Otherwise, the student's solution for that activity was classified as "Incorrect". This code analysis was performed automatically using a Python script that compared the result produced by all the solutions with the expected result.

That analysis was performed for solutions developed by students from both groups. To test the statistical significance of the difference between both groups, the Two Sample t-test was performed when the data is normally distributed (Shapiro-Wilk test with p-value > 0.05), and the Wilcoxon Rank Sum test when the data is non-normally distributed (Shapiro-Wilk test with p-value < 0.05).

To answer the second question, we analyze the data from a post-camp survey question that asks about students' motivation regarding the immediate feedback system and a follow-up question on the post-camp interview. These questions were asked only for students from Group A.

## 6 RESULTS

### 6.1 RQ1: Correctness

To answer the first research question, we present the results of the analysis of the solutions, where the students' solutions for each activity were classified as "Correct" or "Incorrect". There were a total of 24 activities for each group. A total of 526 solutions were submitted by students from Group A, with an average of 21.9 (SD = 1.2). A total of 565 solutions were submitted by students from Group B, with an average of 23.5 (SD = 2.4).

Figure 4 shows the percentage of correct solutions for both groups. The percentage of students that reached a correct solution is normally distributed for both groups, Group A (p-value = 0.2117) and Group B (p-value = 0.1645). The difference in the percentage of correct solutions between groups A and B is statistically significant (Two Sample t-test - p-value = 0.0002145).

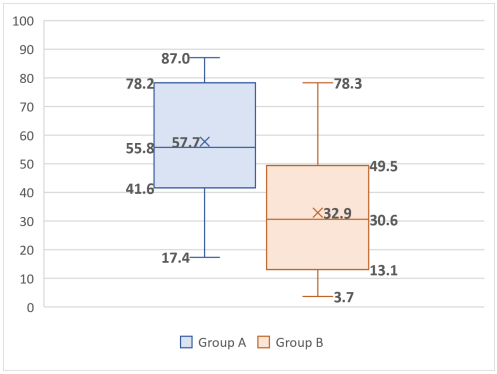


**Figure 4: Percentage of Correct Solutions**

Considering only students from Group A, on average, 59.4% of the students that reached the correct solution were able to do that without receiving any error flagging during the solving process of the activity that they got right. (SD = 21.82; Shapiro-Wilk test p-value = 0.7625).

Analyzing only the long activities, the difference is more expressive. The average percentage of students that got the correct solution is 45.7% for Group A (SD = 15.7) and 13.5% for Group B (SD = 9.4). This difference is statistically significant (Wilcoxon Rank Sum test - p-value = 0.002742).

### 6.2 RQ2: Students Perception

To answer the second research question, we present in Figure 5 the results of students' responses regarding their motivation for each message type. With examples of each message type, the student answered if they felt less motivated, more motivated, or the motivation did not change when seeing a message like the one showed them as an example.

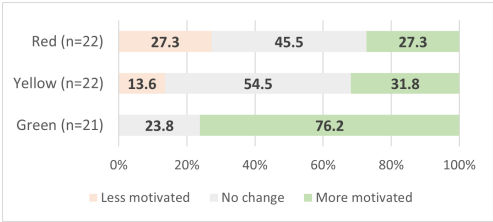


**Figure 5: Students Motivation - Feedback System**

After the camp, Group A students were asked, "how did you know if you coded something correctly or not." One student related the whole feedback system behavior, since the error identification to the indication when the problem was fixed: *"Well, there's something that pops up as an error. It's at the bottom, of yourself, and then you have to read what the problem is or it'll tell you the problem. Then when you find the problem, you could fix it and then it would just go away. I guess that would be how it works."*

Another student pointed out the error flagging functionality that tells the student where the problem is: *"It would tell me the line that was in or row. It would be like, "Error in line 36," or something. Then I'd go to 36 and see what the problem is, or there would be where it would say an error, but it isn't actually an error. You just have to continue the code."*

Finally, one student reported the verification behavior of the feedback system, making a correlation with a check mark: *"It basically gives you this thing on the bottom part where it gives you like a check mark, if it's correct. And if it has anything bad on it, then it says invalid syntax or something."*

Furthermore, they were prompted to discuss the feedback they received, how helpful they were in identifying and fixing mistakes, and what would happen if TunePad had no feedback available.

One student pointed out how the feedback helped him/her to identify the MIDI numbers that were expected: *"The hints were helpful when you were trying to figure out, when you were trying to... when you knew what we wanted it to sound like, but you had to find out which number on the keyboard or which string on the guitar it was. And the hints' kind of helped you lean towards the area that the number you wanted were."*

Some students highlighted the importance of the feedback to help develop a good-sounding solution, for example: *"If they weren't there, then the music would either sound very bad or not play at all."*, and: *"Then until then when you finished the music, you're wondering, "How come it's not working," or it sounds so off and you would have never found it because the error wouldn't have showed you."*

Another student mentioned how hard it would be to find the lines with error if the feedback system was not in use: *"It would've been a lot harder [without the hints]...Because you don't know what you did wrong, so you would have to check every single line of code you have to see what you did wrong. Because it tells you where exactly you did it wrong."*

Finally, one student states that it would not be possible to know what happened with its solution: *"You would never know what happened."*, and another student thinking of a bigger solution stated that with more lines of code the situation would be worst: *"If you had larger lines that go up to 100, then we would be doomed."*

## 7 DISCUSSION

The results presented in this paper provide evidence that using domain-specific immediate feedback improves the percentage of correct solutions developed by the students. The data shows a statistically significant difference in the percentage of students that reached the expected solution between both groups. Students who received domain-specific immediate feedback have a higher percentage of correct solutions than those who did not receive the feedback.

> The percentage of students that reached the correct solution is greater for the group that received the immediate feedback than for the group that did not receive it.

This finding differs in parts from the one reported by Reis et al. [22], where students using the feedback system and only test cases got similar scores in a post-test. It is crucial to mention that the measure used on that work differs from the measure we are using, as we are using specifically the activity correctness and not a post-test. Conversely, Marwan et al. [17], found suggestive evidence that using their feedback system improved students' performance and learning. Finding reinforced by Marwan, Williams, and Price [18], which found evidence suggesting that code hints with textual explanations and code hints with both textual explanations and self-explanations prompts significantly improve performance.

Furthermore, when we isolate and analyze only the long activities, this difference between the averages of the percentage of students that reached the correct solutions from both groups is even higher, indicating that the feedback is even more critical with longer tasks, showing that students are more engaged in doing the activities. The difference in student engagement aligns with findings from Marwan et al. [17], where students using their feedback system significantly improved their engagement. However, it is essential to mention that the measure used in that work differs from the measure we are using.

The results presented in this paper show that students were more motivated by the "green" messages that indicated that they met one requirement. Additionally, the motivation did not change when students saw a "yellow" or "red" message. This suggests that, regardless of the hints and feedback indicating something was wrong, the students were motivated to see that their solution was correct.

> Most students were motivated to receive the "green" feedback during the activity solution process.

This finding is in some measure related to the result reported by Mitrovic, Ohlsson, and Barrow [20], not associated with the motivation but with the importance of the positive feedback, as they report the impact of the positive feedback on students' performance on SQL activities. Moreover, this finding is related to the affective consequences of feedback, where positive feedback increases motivation, carrying information about one's accomplishments, strengths, and correct responses [4].

Analyzing the interview answers, it is possible to conclude that students understood how immediate feedback works and how to use it to reach the objectives of the activities. Furthermore, students believe the feedback is helpful, and without it, it would be much harder to solve the activities.

> Students found the hints and feedback helpful, and it would be much harder to solve the problems without them.

This finding is in line with what is related by Reis et al. [22], where students score their hints system as more useful than test cases, and the students could easily find the location of the errors using the hints system.

## 8 LIMITATION

Due to the nature of the summer camp organization, where each student chooses to register for the session that best suits their interests, it was impossible to have an equivalent distribution of participants between both groups or randomize the students' selection, that is the main reason to use a quasi-experimental setup rather than a controlled experiment. This immediate feedback system was implemented specifically in TunePad. Therefore, the results here cannot necessarily transfer to music programming with other tools; however, TunePad is similar to most other music coding tools.

## 9 CONCLUSION

This paper report results from a summer camp where foundational concepts of computer programming were taught using music. Using a quasi-experimental setup during this camp, we tested an expert-authored domain-specific immediate feedback system comparing the correctness of in-class activities and student perception. The main contributions of this paper aiming to help the CS education community provides evidence that: 1) the use of domain-specific immediate feedback helps students reach the correct solution in a coding activity; 2) the use of positive feedback messages motivates students during a coding activity; and 3) the domain-specific immediate feedback made the solving process easier.

## ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2048792 and 2048793.

# REFERENCES


[1] Samuel Aaron and Alan F. Blackwell. 2013. From Sonic Pi to Overtone: Creative Musical Experiences with Domain-Specific and Functional Languages. In *Proceedings of the First ACM SIGPLAN Workshop on Functional Art, Music, Modeling & Design* (Boston, Massachusetts, USA) *(FARM '13)*. Association for Computing Machinery, New York, NY, USA, 35–46. https://doi.org/10.1145/2505341.2505346

[2] Luciana Benotti, Federico Aloi, Franco Bulgarelli, and Marcos J. Gomez. 2018. The Effect of a Web-Based Coding Tool with Automatic Feedback on Students' Performance and Perceptions. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 2–7. https://doi.org/10.1145/3159450.3159579

[3] Galina Deeva, Daria Bogdanova, Estefanía Serral, Monique Snoeck, and Jochen De Weerdt. 2021. A review of automated feedback systems for learners: Classification framework, challenges and opportunities. *Computers & Education* 162 (2021), 104094. https://doi.org/10.1016/j.compedu.2020.104094

[4] Ayelet Fishbach and Stacey R Finkelstein. 2012. How feedback influences persistence, disengagement, and change in goal pursuit. *Goal-directed behavior* (2012), 203–230.

[5] Jason Freeman, Brian Magerko, Tom McKlin, Mike Reilly, Justin Permar, Cameron Summers, and Eric Fruchter. 2014. Engaging Underrepresented Groups in High School Introductory Computing through Computational Remixing with EarSketch. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (Atlanta, Georgia, USA) *(SIGCSE '14)*. Association for Computing Machinery, New York, NY, USA, 85–90. https://doi.org/10.1145/2538862.2538906

[6] Alex Gerdes, Bastiaan Heeren, Johan Jeuring, and L Thomas Van Binsbergen. 2017. Ask-Elle: an adaptable programming tutor for Haskell giving automated feedback. *International Journal of Artificial Intelligence in Education* 27, 1 (2017), 65–100.

[7] Jamie Gorson, Nikita Patel, Elham Beheshti, Brian Magerko, and Michael Horn. 2017. TunePad: Computational thinking through sound composition. In *Proceedings of the 2017 Conference on Interaction Design and Children*. 484–489.

[8] Sumit Gulwani, Ivan Radiček, and Florian Zuleger. 2018. Automated clustering and program repair for introductory programming assignments. *ACM SIGPLAN Notices* 53, 4 (2018), 465–480.

[9] Philip J Guo. 2013. Online python tutor: embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 579–584.

[10] Michael Horn, Amartya Banerjee, Melanie West, Nichole Pinkard, Amy Pratt, Jason Freeman, Brian Magerko, and Tom McKlin. 2020. TunePad: Engaging learners at the intersection of music and code. (2020).

[11] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. 2018. A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. *ACM Trans. Comput. Educ.* 19, 1, Article 3 (sep 2018), 43 pages. https://doi.org/10.1145/3231711

[12] Irene Lee, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith, and Linda Werner. 2011. Computational Thinking for Youth in Practice. *ACM Inroads* 2, 1 (Feb. 2011), 32–37. https://doi.org/10.1145/1929887.1929902

[13] Colleen M Lewis, Ken Yasuhara, and Ruth E Anderson. 2011. Deciding to major in computer science: a grounded theory of students' self-assessment of ability. In *Proceedings of the seventh international workshop on Computing education research*. 3–10.

[14] Douglas Lusa Krug, Edtwuan Bowman, Taylor Barnett, Lori Pollock, and David Shepherd. 2021. *Code Beats: A Virtual Camp for Middle Schoolers Coding Hip Hop*. Association for Computing Machinery, New York, NY, USA, 397–403. https://doi.org/10.1145/3408877.3432424

[15] Brian Magerko, Jason Freeman, Tom McKlin, Scott McCoid, Tom Jenkins, and Elise Livingston. 2013. Tackling Engagement in Computing with Computational Music Remixing. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 657–662. https://doi.org/10.1145/2445196.2445390

[16] Bill Manaris, Blake Stevens, and Andrew R Brown. 2016. JythonMusic: An environment for teaching algorithmic music composition, dynamic coding and musical performativity. *Journal of Music, Technology & Education* 9, 1 (2016), 33–56.

[17] Samiha Marwan, Ge Gao, Susan Fisk, Thomas W. Price, and Tiffany Barnes. 2020. Adaptive Immediate Feedback Can Improve Novice Programming Engagement and Intention to Persist in Computer Science. In *Proceedings of the 2020 ACM Conference on International Computing Education Research* (Virtual Event, New Zealand) *(ICER '20)*. Association for Computing Machinery, New York, NY, USA, 194–203. https://doi.org/10.1145/3372782.3406264

[18] Samiha Marwan, Joseph Jay Williams, and Thomas Price. 2019. An Evaluation of the Impact of Automated Programming Hints on Performance and Learning. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (Toronto ON, Canada) *(ICER '19)*. Association for Computing Machinery, New York, NY, USA, 61–70. https://doi.org/10.1145/3291279.3339420

[19] Jessica McBroom, Irena Koprinska, and Kalina Yacef. 2021. A Survey of Automated Programming Hint Generation: The HINTS Framework. *ACM Comput. Surv.* 54, 8, Article 172 (oct 2021), 27 pages. https://doi.org/10.1145/3469885

[20] Antonija Mitrovic, Stellan Ohlsson, and Devon K. Barrow. 2013. The effect of positive feedback in a constraint-based intelligent tutoring system. *Computers & Education* 60, 1 (2013), 264–272. https://doi.org/10.1016/j.compedu.2012.07.002

[21] Susanne Narciss. 2008. Feedback strategies for interactive learning tasks. In *Handbook of research on educational communications and technology*. Routledge, 125–143.

[22] Ruan Reis, Gustavo Soares, Melina Mongiovi, and Wilkerson L. Andrade. 2019. Evaluating Feedback Tools in Introductory Programming Classes. In *2019 IEEE Frontiers in Education Conference (FIE)*. 1–7. https://doi.org/10.1109/FIE43999.2019.9028418

[23] Mary Catherine Scheeler, Kathy L Ruhl, and James K McAfee. 2004. Providing performance feedback to teachers: A review. *Teacher education and special education* 27, 4 (2004), 396–407.

[24] Allison Scott, Alexis Martin, Frieda McAlear, and Tia C. Madkins. 2016. Broadening Participation in Computer Science: Existing Out-of-School Initiatives and a Case Study. *ACM Inroads* 7, 4 (Nov. 2016), 84–90. https://doi.org/10.1145/2994153

[25] Valerie J Shute. 2008. Focus on formative feedback. *Review of educational research* 78, 1 (2008), 153–189.