



Engaging K-12 Students with Flow-Based Music Programming: An Experience Report on Its Impact on Teaching and Learning

Zifeng Liu
University of Florida
Gainesville, FL, USA
liuzifeng@ufl.edu

Shan Zhang
University of Florida
Gainesville, FL, USA
zhangshan@ufl.edu

Maya Israel
University of Florida
Gainesville, FL, USA
misrael@coe.ufl.edu

Robert Smith
University of Florida
Gainesville, FL, USA
rpatrick.smith@ufl.edu

Wanli Xing
University of Florida
Gainesville, FL, USA
wanli.xing@coe.ufl.edu

Victor Mincses*
University of California, San Diego
San Diego, CA, USA
vmincses@ucsd.edu

Abstract

Music and computer science (CS) have profound historical and structural connections, with programming music offering a promising avenue for engaging children in CS through creative expression. To foster this engagement, our team developed M-Flow, a flow-based music programming platform designed to introduce students to CS via music. Despite extensive existing research in music and CS education, experience reports and empirical studies on K-12 teachers' implementation and its impact on young kids' learning are limited. Therefore, we recruit elementary school teachers and students with no or limited prior programming experience, introducing them to M-Flow and its curriculum through a professional development workshop, a semester's job embedded support, and classroom implementation. We describe the experiences of teachers as they attempt to integrate music and CS, the challenges they face, and the influence on students' attitudes toward learning computing concepts. Specifically, we reflect on our intervention by conducting a sequential mixed-method evaluation. During the qualitative phase, we collected multiple sources of data from three teachers through focus groups and debriefings after a semester of classroom implementation. Thematic analysis of workshop activities, interviews, and debrief videos revealed three themes with seven sub-themes on teachers' integration of flow-based music programming and two themes with five sub-themes on challenges faced by the teachers. In the quantitative phase, we gathered data on attitudes and self-efficacy from 75 students taught by these teachers. Results indicate that the flow-based music programming environment provided an engaging programming experience for students and significantly increased their self-efficacy towards learning programming.

CCS Concepts

• General and reference → Empirical studies.

Keywords

Flow-based programming; Music; Programming education; K-12; Primary school; Self-efficacy; Teacher experience

ACM Reference Format:

Zifeng Liu, Shan Zhang, Maya Israel, Robert Smith, Wanli Xing, and Victor Mincses*. 2025. Engaging K-12 Students with Flow-Based Music Programming: An Experience Report on Its Impact on Teaching and Learning. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE TS 2025)*, February 26-March 1, 2025, Pittsburgh, PA, USA. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3641554.3701902>

1 Introduction

Computer science (CS) education is evolving towards more creative and interdisciplinary approaches [26]. One of these approaches is music-integrated CS education, which fosters creative thinking and highlights music's structural parallelisms with computer programming [38]. Music education and CS education can be mutually beneficial if well-designed and implemented leveraging each discipline in a pedagogically deliberate manner. Music can make programming activities more engaging, enhancing learners' motivation and participation, while programming can help students better understand musical compositions [16, 17]. As such, studies explored the intersection of computational thinking (CT) and music and emphasized the use of sound as a medium for students to learn CT skills [5, 6].

Previous research has focused on integrating text-based programming languages like Python and Java with music [46]. These programming languages can be challenging for K-12 students, especially primary school students, as they often require a high level of abstraction and programming knowledge that may not be suitable for their developmental stage [13]. In K-12 CS education, students are often encouraged to begin with visual programming practices like flowcharts before actual coding on computers. Research using a Scratch-like platform for middle school students encountered substantial difficulties in maintaining engagement without specialized support [45]. Despite Scratch's simple design, that study showed that significant assistance was needed to help students understand coding.

Thus, we proposed to introduce programming to teachers and students through an interdisciplinary approach, integrating flow-based techniques and music (Figure 1 shows programs created by students using flow-based music programming. These programs



This work is licensed under a Creative Commons Attribution International 4.0 License.

SIGCSE TS 2025, February 26-March 1, 2025, Pittsburgh, PA, USA
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0531-1/25/02
<https://doi.org/10.1145/3641554.3701902>

reflect sequences, parallel procedures, and loops). Flow-based Programming (FBP) is a visual programming paradigm in which processes are represented as boxes connected by arrows [33]. Its flowchart-like architecture simplifies the transition for students from conceptualizing their ideas to converting them into executable code [15]. Thus, it facilitates the rapid development of functional applications by individuals with limited CS background [1, 22].

Several studies have shown that professional development (PD) workshops alongside job-embedded support, can positively impact teachers' perceptions, technical knowledge, and attitudes towards integrating CS and CT into existing disciplines [21, 35]. Our previous studies have explored the impact of a flow-based music programming environment on students, particularly how it enhances their attitudes and engagement with programming [31, 42]. However, there is still a lack of empirical research on how teachers implement CS integration lessons in their classrooms, the challenges they encounter, and the impact of this integration on students' perceptions and attitudes toward learning computing concepts and related subjects. Elementary school teachers typically have no or limited experience with CS and can feel insecure about implementing CS activities [25, 34]. It is crucial to understand how teachers implement flow-based music programming to reach children and investigate the potential impact of a music-making FBP platform on students. This work expands the research to the teacher side.

In this report, we describe our experiences engaging teachers and students with no or limited prior programming experience through a flow-based music programming environment. We introduced a platform called M-Flow and an associated ten-lesson curriculum, with the intervention spanning from November 2023 to May 2024. The curriculum included classroom programming activities where children worked in groups, using M-Flow to create sounds and learn basic programming concepts. We evaluated the intervention from both the teacher and student aspects by conducting a sequential mixed-method evaluation. To understand how teachers integrate flow-based music programming into classrooms and what challenges they face, we collected multiple sources of qualitative data from three teachers, including workshop videos, focus group videos, and debriefing videos after a semester of classroom implementation. We also gathered data on attitudes and self-efficacy from 75 students taught by these teachers to see what impact flow-based music programming has on students. This report offers valuable insights and references for the broader implementation of programming education in K-12 settings, particularly at the primary school level.

2 Related Work and Background

2.1 Integrating CS into STEAM Education

In recent years, integrating CT and CS education into the primary school curriculum has gained significant attention, particularly through STEM integration. Preparing students at an early age for CS education is crucial as early exposure can foster interest and foundational skills essential for future learning [2]. Instead of introducing CS as a new, required discipline, researchers and elementary educators often explore ways to weave computing concepts into existing disciplines. This integrated curriculum development is collaborative, with CT concepts and practices often incorporated into science and mathematics [7, 47]. While most curricula integrate CS

into STEM education, incorporating CS into music and art for K-12 students has shown promising results in creating a more inclusive and engaging learning environment [48]. Among the various approaches in K-12 Arts+CS education, musical flowcharts are a great method to teach computing concepts such as sequences, loops, and conditionals. Through musical flowcharts, students can make connections between different musical sections and understand how they converge to convey a song's message, enhancing students' CT skills, including decomposition, pattern recognition, generalization, and abstraction [16].

2.2 Perceptions of Teachers and Students Toward Integrating CS into Learning

Besides investigating the impact of integrating CS and CT into existing disciplines on student outcomes, it's also important to investigate teachers' experiences and perceptions toward using such tools. Studies have examined teachers' perceptions of integration through PD workshops. For instance, a study [39] investigated elementary teachers' perceptions of integrating CT into their math and science teaching. Despite widely acknowledging challenges such as limited exposure to CS, time constraints, and the difficulties of addressing high-level CT thinking in developmentally appropriate ways, teachers were able to draw strong connections between CT and both mathematics and science instruction. Similarly, [7, 8] found that K-8 teachers achieved significant gains in technical knowledge and held a positive attitude regarding confidence and motivation toward CS and its integration after receiving support and resources in PD workshops. However, there remains an empirical gap in investigating how teachers implement CS with music integration in their classrooms, the challenges they face, and how this integration affects students' perceptions of and attitudes toward learning computing concepts and subject matter.

3 Methods

3.1 M-Flow: Flow-based Music Creation

M-Flow is a FBP tool developed by our team for creating music and sound compositions. Shown in Figure 1, users can drag blocks into the canvas, insert sound recordings into them, connect them with arrows to form sequences, loop them to generate rhythms and manipulate them in various ways. M-Flow offers three different difficulty levels, each featuring distinct functional blocks. In total, there are eight different blocks. An initial version of M-Flow was made public in July 2022. As of July 1, 2024, over 890 teachers and students have registered for accounts on the M-Flow platform. More information can be found on the M-Flow website and other papers describing the tool [32, 42].

3.2 Curriculum

A ten-lesson M-Flow curriculum, aligned with the Computer Science Teachers' Association's (CSTA) standards for CS, has been developed based on the M-Flow platform in collaboration with the Chula Vista Unified School District. The current version of the curriculum is available on our team's [website](#). For details about the curriculum development process, see [42].

The curriculum familiarizes students with the activities and roles of computer programmers and engineers (lesson 1 and lesson 2). It also integrates experiential learning, enabling students to reflect on the creation of sounds and their capacity to convey emotions. Students engage in group collaboration during classroom activities, using M-Flow to explore its various functionalities (lesson 3 to lesson 6). After each exploration session, the class reconvenes for discussions to share observations and elucidate the components of M-Flow. This exploratory pedagogical approach grants students greater autonomy over their learning process. M-Flow’s tiered structure supports an iterative cycle of exploration and discussion, gradually introducing more advanced functionalities and complexities at each level. As students master M-Flow’s features, they create sound compositions designed to express specific emotions (lesson 7 to lesson 10). Finally, students prepare a presentation showing and explaining their work, their creative choice, and their programs’ structure.

3.3 Participants

We recruited both teachers and students for this study. Table 1 shows the teacher participants’ information and students in their class. Three fourth-grade teachers from the western United States participated. The three teachers are women and Latina. Despite their extensive teaching experiences, they have not been involved in teaching CS and possess only a rudimentary understanding of programming concepts. The student population of the school is largely Latinx (88%), with 48% of English Language Learners (ELLs) and 74% of low-income children. 75 fourth-grade students from the three teachers’ classes participated in this study. Ethical approval was received from the Institutional Review Board of the University of Florida.

Table 1: Participants Information

Teacher	Gender	Race	Experience (years)	Subject area	# Students
T1	Woman	Latina	13	Science, Math and Reading	29
T2	Woman	Latina	11	English, Reading, Math and Writing	28
T3	Woman	Latina	10	Reading, Writing, Math, and Science	18
Total students					75

3.4 Procedure

The study was conducted from November 2023 to May 2024 and comprised four stages: workshop, support, classroom implementation, and post-classroom debrief.

On November 29, 2023, three teachers participated in a six-hour in-person workshop at the district office to learn and discuss using M-Flow for classroom instruction. Before the workshop, we collected teachers’ experiences using questionnaires. The workshop participants included the three teachers, the district’s science coordinator, and a member from our team. The workshop aimed to provide a comprehensive understanding of M-Flow and its curriculum, integrating best practices for PD [28]. In the workshop, teachers experienced the curriculum as students, and then reflected on it from the teachers’ perspective, discussing how to implement it in their classrooms.

A week later, we conducted a one-hour focus group to gather teachers’ perceptions of using M-Flow for teaching programming and their classroom implementation plans.

In the second stage, from December 2023 to February 2024, the three teachers, based on the workshop learnings and discussions, finalized preparations for classroom implementation. Our team provided remote support regarding curriculum content queries. During the classroom implementation phase from March to May 2024, two researchers from our team visited the school during lesson 7 to conduct classroom observations and provide support.

The classroom implementation lasted from March to May 2024, the three teachers conducted weekly 45-minute lessons using the M-Flow curriculum. 75 students participated in this stage exploring M-flow and learning the curriculum. Before the classroom implementation, students filled out a pre-survey. After the implementation, students finished a post-survey. Figure 1 shows M-flow programs from students in this lesson.

In the final stage, the three teachers, the district’s science coordinator, and two additional members from our research team conducted a post-classroom implementation discussion on teaching and did a final debrief. This discussion lasted two hours and 30 minutes.

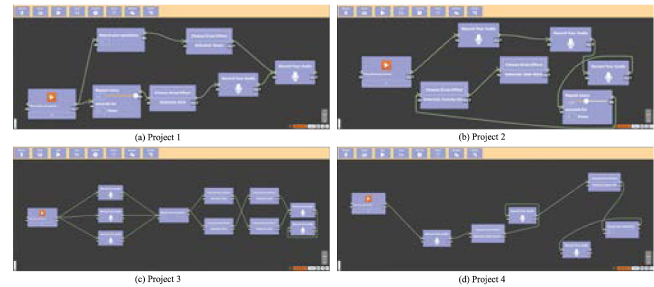


Figure 1: M-Flow projects from students

Note: The four screenshots are from four students’ lesson 7 artifacts created using M-Flow. These programs demonstrate sequences, parallel procedures, and loops.

3.5 Data Collection and Analysis

3.5.1 Instruments. (1) Teacher pre-workshop questionnaire: For teachers, we used an open-ended questionnaire prior to the workshop to gather information on their teaching experience and attitudes. The questions include inquiries about their personal experience with computer programming and their familiarity and practical engagement with programming. (2) Teacher focus group questions: The focus group questions are adapted from [40] and were categorized into five main areas: Epistemology, Self-efficacy, Growth Mindset, Creativity, and Outcome. It contains questions like “How do you feel about using the flow-based music creation (M-Flow) platform? Why?” (3) Students pre and post-survey: We implemented a scale to collect data on students’ experiences, self-efficacy, interest and willingness, and identity-related to engineering and programming. Table 2 details the questions used, which cover four constructs: experience, self-efficacy, interest and willingness, and CS identity. Items on self-efficacy, willingness to persist in learning, and CS identity are adapted from [29] and [36].

Table 2: Student Survey Items

Category	Item #	Question (scale)
Experience	1	Do you know what programming a computer means? (1-3)
	2	How often did you program a computer in the past? (1-3)
	3	I have programmed a computer to make music and sounds (1-5)
Self-efficacy	4	I am able to learn to program computers to make music and sounds (1-5)
	5	I have programmed computers to do things I want (1-5)
	6	I am able to learn to program a computer to do things I want (1-5)
Interest and Willingness	7	If my middle school offers computer programming classes, I would like to sign up (1-5)
	8	If my school offers, for making music I would sign up (1-5)
Identity	9	When I grow up, I would consider working as a computer engineer (1-5)
	10	When I grow up, I want to use programming in my job (1-5)

Note: Item 1 has options No (1), Kind of (2), and Yes (3). Item 2 has options I never programmed (1), I programmed a little (2), and I programmed a lot (3). Items 3 to 10 use a five-Likert scale from 1 (Absolutely no) to 5 (Yes).

3.5.2 Data Collection. To evaluate the outcome of flow-based music programming on teaching and learning, we utilized the instruments mentioned above as well as video recordings for data collection. For teachers, in addition to pre-workshop questionnaires and focus group (one hour), we recorded a four-hour workshop video and a two-hour-and-twenty-minute post-classroom implementation debrief video for analysis. For students, pre- and post-survey data were collected on their experience, self-efficacy, interest and willingness, and identity towards engineering and programming.

3.5.3 Data Analysis. For teachers' qualitative data, we used thematic analysis to understand how they implement the curriculum in the classroom and what challenges they have. Thematic analysis is a broad, encompassing approach to uncovering recurring patterns within qualitative data [3]. We use this approach as it allows for an in-depth exploration of teachers' experiences, attitudes, pedagogical strategies, and challenges in the intervention. For students' survey data, we used descriptive analysis and t-tests to measure if there was a change in students' attitudes and self-efficacy.

4 Results & Findings

4.1 Teachers' Data Results

We analyzed the workshop video (four hours), the focus group (one hour), and the debrief videos (two hours and 20 minutes) using thematic analysis to find out how teachers integrate M-Flow into their classrooms and the challenges they face. We identified 104 relevant script pieces with 17 different codes. After refining, aggregating, and removing codes unrelated to the integration, we found three themes with seven sub-themes for teacher integration of flow-based music programming and two themes with five sub-themes for challenges teachers face in classroom implementation.

4.1.1 Theme 1: Teaching Strategies. The teachers applied various teaching strategies to integrate flow-based music programming into their classrooms. One of these strategies was collaborative teaching and learning (sub-theme 1.1). For instance, teachers T1 and T2 mentioned that they would collaborate on preparing lesson materials and share their experiences: "After we've done a few lessons, we'll kind of know the rhythm of it. And then you and I can talk about it like this is where we're at." (T2, workshop) Besides, the teachers tried to create a collaborative learning environment for students. The teachers paired all students for programming tasks and clearly

defined roles within the collaboration (T3, debrief). They also provided specific times for role switching, which effectively helped students complete their learning tasks: "We'll partner students up and assign roles. One student will be the driver and the other will be the director." (T3, workshop)

Teachers also planned and successfully ensured a student-centered learning environment (sub-theme 1.2). The teachers viewed themselves as facilitators by providing clear direction and exceptions, allowing students to explore independently. For example, T3 said in the interview that *I see myself as a facilitator, as a facilitator, I aim to be clear and direct with expectations and directions from the beginning. I will not only provide these verbally but also display them clearly.* This ensured students' active engagement when using M-Flow in the classroom.

4.1.2 Theme 2: Scaffolding and Guidance. Another theme focused on Scaffolding and Guidance. This theme includes diverse learner support (sub-theme 2.1) and structured support (sub-theme 2.2). The first aspect addresses the needs of special students to enhance their engagement with M-Flow. Additionally, from the teachers' perspective, the second aspect reflects on the importance of structuring support and direction provided to students.

4.1.3 Theme 3: Assessment method. The teachers mentioned using multiple methods to assess students' grasp of programming knowledge in the interview and debrief, including rubrics and checklists (sub-theme 3.1), verbal and reflective assessments (sub-theme 3.2), and demonstrations and presentations (sub-theme 3.3): "I like your idea about getting the kids brought up to their computer and showing you how they came up with a loop. Even though we have the videos, show what the kid is doing and then show the actual." (T2, debrief)

4.1.4 Theme 4: Student-related challenges. Other challenges we found are student-related, including student engagement (sub-theme 4.1), exploration (sub-theme 4.2), and classroom management (sub-theme 4.3). In the workshop, teachers expressed concerns about certain students, such as language learners or students with disabilities. For example, they worried about how these students would engage with and explore the M-Flow platform effectively: "my students who with a disability, at least for my class, they just kind of don't participate as much or they're just kind of in their own world." (T2, workshop).

However, in the debrief, teachers did not mention concerns about student engagement. Instead, they discussed challenges students faced during independent exploration, such as difficulty understanding tasks, coming up with solutions, and handling multiple tasks simultaneously: "Some students were really focused on that emotion. So they're trying to figure out a sound that goes with it. They were just kind of walking around and couldn't come up with one." (T2, debrief).

4.1.5 Theme 5: Pedagogical challenges. The teachers also faced pedagogical challenges. Due to their limited knowledge of CS and programming (sub-theme 5.1), they were concerned about their ability to answer students' questions and provide deeper instruction. For example, T3 mentioned in the debrief that *I was kind of lost in that translation where all of a sudden the kids were doing a lot more and like what would just happen there? So I just kind of like I didn't know. So I'm like, alright, you figure it out.*

Another pedagogical challenge the teachers described as the tension between planning strategies aimed at promoting both scaffolds for some learners and student creative expression (sub-theme 5.2). For example, teachers used scaffolds such as sentence frames to help ELLs express themselves in class, but they indicated that this support may have limited the creative expression of other students, leading teachers to question the necessity of such strategies. T2 said in the debrief that *"I don't know if the frames themselves provide that authentic conversation or lend themselves to this type of lesson, what we're doing a reading lesson or something and you're talking about a text, it's easier to kind of refer to a sentence. But this (programming learning) is a little bit different, because everyone's doing the discretions are different."* This issue also appeared in pair programming, where in some pairs, one child took on all the exploration tasks while the other completely stepped back.

4.2 Students' Data Results

The overall description column in Table 3 shows the survey results from students who completed the survey. According to the pre-survey results, 39% of the children responded that they did not have any programming experience, and 46% said they had minimal programming experience, resulting in a total of 85% of participants with no or minimal prior experience. Data from 61 students could be matched between the pre- and post-survey.

The paired t-test in Table 3 is from students who answered *Kind of* or *Yes* to item #1 in the pre-survey, indicating that they know what programming means (n=36). We can see a significant increase in students' experience (#1 to #3) and self-efficacy (#4 to #6) after the classroom implementation. However, there was no significant change in interest and willingness, nor in the identity score.

Table 3: Pre and Post comparison by categories

Category	Item	Overall description		Paired t-test		One sample t-test	
		# of students	Mean(std)	T statistics (n=36)	Effect size	T statistics (n=25)	Effect size
Experience	#1	Pre (n=69)	1.75 (0.69)	2.62 *	0.54	14.51**	4.10
		Post (n=69)	2.46 (0.55)				
	#2	Pre (n=37)	1.84 (0.49)	3.95**	0.83	4.80**	1.36
		Post (n=69)	2.25 (0.60)				
	#3	Pre (n=35)	3.11 (0.98)	7.76**	1.57	9.56**	2.70
		Post (n=69)	4.41 (0.69)				
Self-efficacy	#4	Pre (n=36)	3.44 (1.12)	4.07**	0.77	5.77**	1.63
		Post (n=69)	4.14 (0.84)				
	#5	Pre (n=36)	3.17 (1.12)	2.39*	0.51	4.68**	1.32
		Post (n=69)	3.88 (1.01)				
	#6	Pre (n=35)	3.66 (1.14)	1.83	0.16	4.74**	1.34
		Post (n=66)	4.02 (1.01)				
Interest and Willingness	#7	Pre (n=41)	3.39 (1.30)	-0.73	-	1.19	-
		Post (n=69)	3.33 (1.14)				
	#8	Pre (40)	3.60 (1.28)	-1.50	-	1.81	0.27
Identity	#9	Pre (n=41)	2.66 (1.16)	0.13	-	-0.96	-
		Post (n=69)	2.81 (1.09)				
	#10	Pre (n=41)	3.15 (1.24)	-1.05	-	0.33	-
		Post (n=69)	3.10 (1.21)				

Note: ** $p < 0.01$, * $p < 0.05$. The overall description shows the results of all the students who completed the survey. For the 36 students who had matched pre- and post-survey data, we conducted paired t-tests. Effect sizes were calculated using Cohen's D. For the 25 students who previously reported not knowing what programming meant, and thus did not have pre-survey answers for survey items #2 to #10, we analyzed their responses using one-sample t-tests to assess whether their attitudes were generally positive by comparing their scores to a neutral value (e.g., 2 on a 1–3 Likert scale).

For children who in the pre-survey indicated that they didn't know what programming meant (n=25), we run a one sample t-test, comparing their scores to a neutral valence. Results show that, after going through the curriculum, they had a positive experience and

high self-efficacy ($p < 0.01$ for #1 to #6). For questions 8 to 10 the results were not significantly different from the neutral value. Figure 1 shows screenshots of projects created by four children in lesson 7. Children created projects with different levels of complexity (e.g., number of nodes and types of blocks). For example, in Figure 1(c), the project is very structured, consisting of 11 blocks, which reflect the concepts of parallelism and loops in computing.

5 Discussions and Implications

Previous research demonstrates that EarSketch [18] and TunePad [14] have been effective in engaging children, although the children were older, and the experiences relied on specialized educators with deep knowledge of both music and programming [37]. Flow-based programming offers an alternative that addresses these challenges. It is designed not only to be accessible to younger students but also to be implemented by non-specialized teachers. This adaptability is a key strength of our method, allowing for broader accessibility without the need for extensive technical training or background knowledge in CS or music. Furthermore, our focus on creating positive, early experiences aligns with research showing the long-term benefits of early exposure to CT and creative problem-solving [43]. Even though scratch [11, 50] has music-making capabilities, limited research has explored its impact on students' attitudes or the extent of teacher training required for Scratch-based interventions. Moreover, a flow-based approach is more suitable for replicating complex musical structures. For example, in Scratch, generating bifurcating parallel structures requires the use of 'broadcasting,' which is relatively advanced. However, in M-flow, a single block can have multiple output arrows that simultaneously generate parallel sound streams, simplifying the process for both students and teachers [42].

5.1 Integration Strategies

The thematic analysis identified three themes regarding teachers' integration of flow-based music programming into classroom teaching. First, teachers employed strategies focused on collaborative teaching and learning, as well as student-centered learning. For their own collaboration, teachers planned to engage in team teaching, including discussions after teaching several lessons (T2, workshop) and dividing instructional responsibilities (T1, workshop). For students' collaborative learning, they are assigned roles such as director and driver (T2, T3 interview), which is beneficial for facilitating pair programming and enabling students to build on each other's ideas [49]. In student-centered learning, teachers acted as facilitators, primarily setting expectations and directions (T1, T3 workshop), allowing students to explore the M-Flow platform independently and engage in peer learning [24].

Regarding assessments, teachers mentioned that they would use rubrics and checklists to evaluate student learning (T1, T2, T3 interviews), along with verbal and reflective assessments by asking students questions (T1, debrief). Additionally, teachers discussed using presentations where students screen-record their work to showcase their learning (T2, debrief). Integrating methods such as reflective assessments and student presentations in programming classes aligns with research underscoring the benefits of diverse assessment techniques. These approaches offer deeper insights into

students' comprehension and application of programming concepts while promoting fairer evaluation practices [12, 27].

5.2 Challenges and Solutions

Some student-related challenges were recognized in this implementation process. Regarding student engagement, teachers expressed concern that some students, such as students with disabilities, might struggle with navigating a computer and might not participate, becoming absorbed in their own world (T2, workshop). Sometimes, students talked to each other but were not effectively communicating (T1, workshop). Additionally, some students faced decision-making difficulties, so teachers aimed to provide options whenever possible (T3, debrief).

Pedagogical challenges include teachers' limited knowledge of programming and M-Flow and restrictions brought by certain scaffolding and guidance strategies. T1 mentioned in the interview that a challenge for her is that students may want to do more with M-Flow than she can actually support. Sometimes, students did much more than the teacher understood, and the teacher had to let them figure it out independently (T3, debrief). However, there is a need for teachers to balance guidance with opportunities for independent learning [30]. A well-designed tool could support students' inquiry while allowing teachers to facilitate effectively [4, 19]. Pair programming, although fostering collaboration, resulted in only one student having access to a computer at a time, and they might not switch roles, causing the other student to disengage (T1, T3 debrief). It noted the importance of role rotation in collaborative learning to ensure active participation from all students [19, 20]. [44] emphasized the need for differentiated instruction to foster creativity. Some strategies used by teachers in this study are critical for certain students. However, teachers could offer other students the choice of using these strategies or not, thereby alleviating the restrictions imposed by these strategies.

5.3 Impact on Students' Attitudes

The study revealed that flow-based music programming significantly enhanced students' programming experience and self-efficacy, particularly those without prior experience. Specifically, 85% of participants had no or minimal prior programming experience, and their experience and self-efficacy scores showed significant improvement after using M-flow. Despite these gains, there were no significant changes in students' interest, willingness, and identity scores. This is consistent with studies that show that career-related changes typically develop gradually, with significant shifts being rare during school years, particularly in STEAM fields [41]. Younger students often have broad, changeable interests and may lack the motivation or capacity to connect current experiences with future career goals [23]. At this stage, fostering positive early experiences might prove more crucial, as these experiences play a foundational role in future development [43].

These findings align with the literature, which suggests that creative programming activities, such as music programming, can effectively build computing skills and confidence among novice learners. For instance, [12] found that high school students' engagement in computational remixing with EarSketch increased interest and attitudes in computing. Similarly, [10] reported that high school

students enhanced their creative expression through significant gains in computing attitudes and creativity while composing and remixing songs in a programming environment, highlighting the positive impact of music-based programming on students' skills and attitudes. Additionally, [14] noted a platform for sound composition, supported CT through music, which increased student interest and engagement and provided the potential to broaden the participants in CS. Despite these positive outcomes, maintaining and enhancing students' long-term interest and identification with computing remains a challenge. Research by [9] indicates that while initial exposure to engaging programming activities can boost confidence and skills, sustained interest requires continued engagement and support. While students feel more competent, additional efforts are needed to sustain their intrinsic motivation.

6 Conclusion

In this report, we explored the impact of flow-based music programming on teaching and learning. After participating in workshops, discussions, and classroom implementations over six months, three teachers with no prior programming teaching experience completed M-Flow classroom instruction for 75 students. We described how teachers employed various strategies to integrate flow-based music programming into their classroom, specifically through collaborative teaching and learning, providing students with scaffolding and guidance, using multiple assessment methods. Student-related and pedagogical challenges were also found in the implementation process. For students, we found that flow-based music programming enhanced students' programming experience and self-efficacy. Future work will refine the curriculum and create a streamlined, PD program to reach more teachers and students, continuing to examine the program's impact on teachers and students.

Acknowledgments

This work was supported by the National Science Foundation's ITEST program (22-585), "Using Flow-Based Music Programming to Engage Children in Computer Science."

References

- [1] Zaw Htet Aung, Soonthareeya Sanium, Chuenchat Songsaksupachok, Worapan Kusakunniran, Monamorn Precharattana, Suparat Chuechote, Khemmawadee Pongsanon, and Panrasee Ritthipravit. 2022. Designing a novel teaching platform for AI: A case study in a Thai school context. *Journal of Computer Assisted Learning* 38, 6 (2022), 1714–1729.
- [2] Marina Umaschi Bers. 2019. Coding as another language: A pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education* 6, 4 (2019), 499–528.
- [3] Virginia Braun and Victoria Clarke. 2019. Reflecting on reflexive thematic analysis. *Qualitative research in sport, exercise and health* 11, 4 (2019), 589–597.
- [4] Su Cai, Zifeng Liu, Chao Liu, et al. 2022. Effects of a BCI-Based AR Inquiring Tool on Primary Students' Science Learning: A Quasi-Experimental Field Study. *Journal of Science Education and Technology* 31 (2022), 767–782. <https://doi.org/10.1007/s10956-022-09991-y>
- [5] Josh Caldwell. 2022. *Coding and the Arts: Connecting CS to Drawing, Music, Animation and More*. International Society for Technology in Education.
- [6] Mário Anibal Cardoso, Elsa Maria Gabriel Morgado, and Levi Leonido. 2023. Unleashing creative synergies: A mixed-method case study in music education classrooms. *Applied Sciences* 13, 17 (2023), 9842.
- [7] Mehmet Celepkolu, Erin O'Halloran, and Kristy Elizabeth Boyer. 2020. Upper elementary and middle grade teachers' perceptions, concerns, and goals for integrating CS into classrooms. In *Proceedings of the 51st ACM technical symposium on computer science education*. 965–970.

- [8] Mehmet Celepkolu, Erin O'Halloran, Jamieka Wilkinson, and Kristy Elizabeth Boyer. 2019. An Analysis of Upper Elementary and Middle Grade Teachers' Perceptions, Concerns and Goals for Integrating CS into Classrooms. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 1257–1257.
- [9] Jill Denner, Linda Werner, and Eloy Ortiz. 2012. Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education* 58, 1 (2012), 240–249.
- [10] Shelly Engelman, Brian Magerko, Tom McKlin, Morgan Miller, Doug Edwards, and Jason Freeman. 2017. Creativity in authentic STEAM education with EarSketch. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. 183–188.
- [11] Deborah Fields, Veena Vasudevan, and Yasmin B Kafai. 2015. The programmers' collective: fostering participatory culture by making music videos in a high school Scratch coding workshop. *Interactive Learning Environments* 23, 5 (2015), 613–633.
- [12] 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*. 85–90.
- [13] Marcos J Gomez, Marco Moresi, and Luciana Benotti. 2019. Text-based programming in elementary school: a comparative study of programming abilities in children with and without block-based experience. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. 402–408.
- [14] 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.
- [15] Cesar Goudouris, Antônio Carlos de Abreu Mol, Ana Paula Legey, Paulo Victor Rodrigues de Carvalho, Joana Loureiro Freire, Bianca Maria Rego Martins, and Alessandro Jatobá. 2020. Applying flow-based principles in teaching computer programming to high school students: A semiotic perspective. *Education and Information Technologies* 25 (2020), 5451–5476.
- [16] Gena R. Greher and Jesse M. Heines. 2014. *Computational Thinking in Sound: Teaching the Art and Science of Music and Technology*. Oxford University Press.
- [17] Michael S. Horn, Michael West, and Cameron Roberts. 2022. *Introduction to Digital Music with Python Programming: Learning Music with Code*. CRC Press.
- [18] Fatemeh Jamshidi, Maryam Bigonah, and Daniela Marghitu. 2024. Striking a Chord through a Mixed-Methods Study of Music-Based Learning to Leverage Music and Creativity to Bridge the Gender Gap in Computer Science. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 1694–1695. <https://doi.org/10.1145/3626253.3635612>
- [19] Xinyue Jiao, Zifeng Liu, Haitao Zhou, and Su Cai. 2022. The Effect of Role Assignment on Students' Collaborative Inquiry-based Learning in Augmented Reality Environment. In *2022 International Conference on Advanced Learning Technologies (ICALT)*. 349–351. <https://doi.org/10.1109/ICALT55010.2022.00109>
- [20] Mina C Johnson-Glenberg, David A Birchfield, Lisa Tolentino, and Tatiana Koziupa. 2014. Collaborative embodied learning in mixed reality motion-capture environments: Two science studies. *Journal of educational psychology* 106, 1 (2014), 86.
- [21] Diane Jass Ketelhut, Kelly Mills, Emily Hestness, Lautaro Cabrera, Jandelyn Plane, and J Randy McGinnis. 2020. Teacher change following a professional development experience in integrating computational thinking into elementary science. *Journal of science education and technology* 29 (2020), 174–188.
- [22] Kodigy. 2022. The state of Flow-based Programming - A system brought to life. <https://blog.kodigy.com/post/flow-based-programming>. Accessed: 2024-01-15.
- [23] Esra Kızılay and Havva Yamak. 2023. Factors Affecting High School Students' Motivation and Career Interest in STEM Fields and Their Modeling. *Science Insights Education Frontiers* 16, 1 (May 2023), 2409–2433. <https://doi.org/10.15354/sief.23.or256>
- [24] Eunbae Lee and M. Hannafin. 2016. A design framework for enhancing engagement in student-centered learning: own it, learn it, and share it. *Educational Technology Research and Development* 64 (2016), 707–734. <https://doi.org/10.1007/S11423-015-9422-5>
- [25] Dan Leyzberg and Christopher Moretti. 2017. Teaching CS to CS teachers: Addressing the need for advanced content in K-12 professional development. In *Proceedings of the 2017 ACM SIGCSE technical symposium on Computer Science Education*. 369–374.
- [26] Zifeng Liu, Rui Guo, Xinyue Jiao, Xueyan Gao, Hyunju Oh, and Wanli Xing. 2024. How AI Assisted K-12 Computer Science Education: A Systematic Review. In *2024 ASEE Annual Conference & Exposition*. Portland, Oregon. <https://doi.org/10.18260/1-2--47532>
- [27] Zifeng Liu, Xinyue Jiao, Chenglu Li, and Wanli Xing. 2024. Fair Prediction of Students' Summative Performance Changes Using Online Learning Behavior Data. In *Proceedings of the 17th International Conference on Educational Data Mining*. International Educational Data Mining Society, Atlanta, Georgia, USA, 686–691. <https://doi.org/10.5281/zenodo.12729918>
- [28] Susan Loucks-Horsley, Katherine E Stiles, Susan Mundry, Nancy Love, and Peter W Hewson. 2009. *Designing professional development for teachers of science and mathematics*. Corwin press.
- [29] Stacie L Mason and Peter J Rich. 2020. Development and analysis of the elementary student coding attitudes survey. *Computers & Education* 153 (2020), 103898.
- [30] Richard E Mayer. 2009. Constructivism as a theory of learning versus constructivism as a prescription for instruction. In *Constructivist instruction*. Routledge, 196–212.
- [31] Victor Hugo Mincez and Nagarajan Akshay. 2023. *STEAM for all: a vision for STEM and arts integration*.
- [32] Victor Hugo Mincez, Wanli Xing, and Chenglu Li. 2023. Work in Progress: Mflow, a Flow-based Music Programming Platform for Young Children. In *2023 IEEE World Engineering Education Conference (EDUNINE)*. IEEE, 1–4.
- [33] J Paul Morrison. 2010. *Flow-Based Programming: A new approach to application development*. CreateSpace.
- [34] E. Novak and J. I. Khan. 2022. A Research-Practice Partnership Approach for Co-Designing a Culturally Responsive Computer Science Curriculum for Upper Elementary Students. *TechTrends* 66 (2022), 527–538. <https://doi.org/10.1007/s11528-022-00730-z>
- [35] Ayodele Abosede Ogebo and Umesh Ramnarain. 2022. Teachers' perceptions of and concerns about integrating computational thinking into science teaching after a professional development activity. *African Journal of Research in Mathematics, Science and Technology Education* 26, 3 (2022), 181–191.
- [36] Outlier Research & Evaluation. 2017. *BASICS Study ECS Student Implementation and Contextual Factor Questionnaire Measures [Measurement scales]*. Technical Report. Outlier Research & Evaluation at UChicago STEM Education | University of Chicago, Chicago, IL. <http://outlier.uchicago.edu/basics/resources/Measures-StudentImplementation/>
- [37] Christopher Petrie. 2023. Design and use of domain-specific programming platforms: interdisciplinary computational thinking with EarSketch and TunePad. *Computer Science Education* (2023), 1–34.
- [38] Alexander Repenning, Jürg Zurmühler, Anna Lamprou, and Daniel Hug. 2020. Computational Music Thinking Patterns: Connecting Music Education with Computer Science Education through the Design of Interactive Notations. In *CSEdu (1)*. 641–652.
- [39] Kathryn M Rich, Aman Yadav, and Christina V Schwarz. 2019. Computational thinking, mathematics, and science: Elementary teachers' perspectives on integration. *Journal of Technology and Teacher Education* 27, 2 (2019), 165–205.
- [40] Darcy Ronan, D Cenk Erdil, and Dennis Brylow. 2023. Teacher attitudes & beliefs in computer science (T-ABC): Development & validation of a teacher survey instrument. *ACM Transactions on Computing Education* 23, 2 (2023), 1–23.
- [41] Nargiza Sharapova, Saule Zholdasbekova, Sholpan Arzymbetova, Omer Zaimoglu, and Gulshat Bozshatayeva. 2023. Efficacy of school-based career guidance interventions: A review of recent research. *Journal of Education and E-Learning Research* 10, 2 (2023), 215–222. <https://doi.org/10.20448/jeelr.v10i2.4554>
- [42] Yukyueong Song, Wanli Xing, Alec Barron, Hyunju Oh, Chenglu Li, and Victor Mincez. 2023. M-flow: a Flow-based Music Creation Platform Improves Underrepresented Children's Attitudes toward Computer Programming. In *Proceedings of the 22nd Annual ACM Interaction Design and Children Conference* (Chicago, IL, USA) (*IDC '23*). 233–238. <https://doi.org/10.1145/3585088.3589383>
- [43] Robert H Tai, Christine Qi Liu, Adam V Maltese, and Xitao Fan. 2006. Planning early for careers in science. *Science* 312, 5777 (2006), 1143–1144.
- [44] Carol Ann Tomlinson. 2022. *Everybody's classroom: differentiating for the shared and unique needs of diverse students*. Teachers College Press.
- [45] Daniel A Walzer and Jesse M Heines. 2019. Teaching a Computer to Sing. *Integrating Digital Technology in Education: School-University-Community Collaboration* (2019), 31.
- [46] Harco Warnars and Winston Rusli. 2021. A literature review of music in computer science. *International Journal of Computing and Digital System* (2021). <https://dx.doi.org/10.12785/ijcds/1201120>
- [47] Kevin P Waterman, Lynn Goldsmith, and Marian Pasquale. 2020. Integrating computational thinking into elementary science curriculum: An examination of activities that support students' computational thinking in the service of disciplinary learning. *Journal of Science Education and Technology* 29, 1 (2020), 53–64.
- [48] Lauren Weisberg, Joanne Barrett, Maya Israel, and Don Miller. 2024. A review of arts integration in K-12 CS education: gathering STEAM for inclusive learning. *Computer Science Education* (2024), 1–30.
- [49] L. Williams, E. Wiebe, Kai Yang, M. Ferzli, and Carol Miller. 2002. In Support of Pair Programming in the Introductory Computer Science Course. *Computer Science Education* 12 (2002), 197 – 212. <https://doi.org/10.1076/csed.12.3.197.8618>
- [50] LeChen Zhang and Jalal Nouri. 2019. A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education* 141 (2019), 103607. <https://doi.org/10.1016/j.compedu.2019.103607>