Informatics in Education, 2023, Vol. 22, No. 1, 45–69

DOI: 10.15388/infedu.2023.06

Leveraging the Pied Piper Effect – The Case of Teaching Programming to Sixth-grade Students Via Music

Ilana LAVY
Yezreel Valley College, ISRAEL
e-mail: ilanal@yvc.ac.il

Received: January 2022

Abstract. This paper describes a pilot study that explores students learning how to program via a multi-disciplinary approach. The study participants were eleven 6th grade students who learned programming fundamentals via music activities in a Scratch 3.0 environment. These activities included the programming of familiar melodies and the development of suitable animations or computer games. For that matter, a study unit termed MelodyCode was developed in the spirit of the STEAM education approach and the spiral learning method and included exploration tasks based on individual learning. Via the programming of familiar melodies, they became acquainted with programming concepts such as functions, variables, repetition and control commands, parallel processes, and more. Competitions that win awards were held from time to time, which prompted students to invest efforts in their projects to reach first place and gain the teacher and classmates' appreciation. The study was conducted in the form of action research. The data analysis yielded references to the effect of MelodyCode on common stereotypes students hold regarding programming (masculine profession, necessitates good mathematics knowledge), cognitive aspects (cognitive load, linking music concrete use to abstract programming concepts), and affective aspects (joyful and relaxing class atmosphere, motivation, curiosity, self-efficacy).

Keywords: music, programming fundamentals, spiral learning, Scratch environment, STEAM education.

1. Introduction

Preparing today's pupils for optimal functioning in the developing technological world of tomorrow necessitates the expanding of the population (males and especially females) of creative people with high cognitive abilities, analytic thinking, technological abilities, and problem-solving skills (Yang *et al.*, 2015). Even though programming is considered a complex topic to perceive by beginner students (Margolis and Goode, 2016; Bosse and Gerosa, 2017), efforts must be made to expand the group of programmers to keep the momentum of meaningful technological development.

Delving into the meaning of programmers' training reveals a resemblance between learning how to program and acquiring a new language. Learning a new language and programming learning are easy to perceive at a younger age (Papadakis *et al.*, 2016). Moreover, in addition to achieving the goal of expanding the programmer population, during the programming studies, the students develop essential skills such as analytical thinking abilities, problem-solving skills, and creativity that might improve their cognitive skills (White and Sivitanides, 2002; Thomas *et al.*, 2007; Papadakis *et al.*, 2016; Kalelioğlu and Gülbahar, 2014). Developing problem-solving skills involves developing high-order thinking skills that help people face cognitive challenges. Through finding an algorithm that solves the problem, skills such as analytic thinking and creativity are required (White and Sivitanides, 2002). According to Neo-Piagetian theories of cognitive development, people, regardless of their age, develop abstract abilities as they gain expertise in a specific domain. Thus, a novice in one domain will exhibit less abstract abilities than that person will exhibit in a domain where he/she is an expert (Morra *et al.*, 2007). The exposure of 6-grade students to programming learning is relatively low. Hence, students of this age can be considered novices. As such, they do not possess the necessary abstract abilities required to understand abstract programming concepts. Having that in mind, a concrete mediation of the programming abstract concepts is required for novice programmers. Within this study, the selected concrete mediation is via music. As described in the Pied Piper legend and stated by researchers, music induces magic in people (Gold *et al.*, 2013; Zhu *et al.*, 2005); music listening frequently has a more immediate outcome than using a program and often affects emotions (Koelsch, 2010; Zentner *et al.*, 2008). Hence, the idea to mediate the abstract programming concepts via exciting concrete music activities was raised. Using music as a mediator serves two purposes, to create a situation in which a need for programming will stem from the students and provide students with immediate outcomes that might increase their motivation to cope with programming.

Music's role is to bring a joyful, concrete meaning to abstract programming concepts. Making the connection between programming and music also stems from embracing the STEAM education approach that integrates Science, Technology, Engineering, Arts, and Mathematics disciplines (or part of them) (Kim *et al.*, 2012). STEAM programs aim to teach students innovation, think critically, and use technology in imaginative designs or creative approaches to real-world problems. It also nurtures curiosity, creativity, and flexibility and fosters self-regulated learning skills (Thomas *et al.*, 2007; Papadakis *et al.*, 2016).

One of the teaching methods that has evolved over the years to handle complex concepts is the spiral method (Bruner, 1960). In this approach, topics or skills are revisited while their level of complexity increases (Takaya, 2008). First, the subject is taught on a basic level, followed by several iterations, adding more complexity to the topic, and reinforcing previously discussed principles. This teaching method facilitates the establishment of the learning topic thoroughly. The spiral learning approach enables a gradual learning process (Johnston, 2012) and is suitable for teaching young students programming principles.

To employ the MelodyCode learning unit the blocks-based Scratch 3.0 environment was chosen. The Scratch environment is already applied in elementary schools in our country but on a small scale. It should be specified that the teaching focus in MelodyCode differs from the methods usually used. Via the programming of familiar songs, a need for learning fundamental programming concepts will rise. In addition, the teaching/learning process follows the constructivist theory so that the students will engage in personal exploration tasks. In the last fifteen minutes of each lesson, a class discussion will take place in which new concepts will be summarized, and students will be able to share their gained insights.

This study is part of a broader study that explores various aspects of learning to program via music on different research populations. Lavy's pilot study (2021) presents findings regarding the effect of using music on learning the basics of programming of seventh- and 8th-grade students. The current pilot study examines whether the above idea is also suitable for younger students. To this end, the study units have been rewritten and modified to fit 6th-grade learning level. The present study examines the effect of learning programming via music on 6th-grade students' perceptions regarding programming studies.

2. Theoretical Background

In what follows, presented a brief theoretical background on the following issues: music for the sake of programming, STEAM education, the spiral curriculum, cognitive load theory, and developing programming self-efficacy.

2.1. *Music for the Sake of Programming*

Music, in its universality, plays a meaningful role in both human society and well-being (VanderArk and Ely, 1991). Music offers a unique perspective on life and on studying as well. The influence of music on people led to the development of legends describing its impact on living beings, such as the Pied Piper of Hamlin. Music engagement nurtures problem-solving, critical, and higher-order thinking skills (Bamberger, 1982; 2000; Powers, 2011). Higher-order thinking, including analysis, synthesis, and evaluation (Bloom, 1956), provides a theoretical basis for critical thought (Topoğlu, 2014; Olson, 2000) that is not domain-specific.

A link between music practice and the development of critical thinking that is not domain-specific was observed (Johnson, 2011; Zellner, 2011; Bamberger, 2000; Olson, 2000). These researchers found that engaging in music requires listening, detecting incorrect musical notes, identifying musical structures, harmonies, and more. These actions help develop skills such as problem-solving and critical thinking. Engaging in music involves thinking about cognitive processes, in which the learners develop their understanding of music and become active problem-solvers (Bell and Bell, 2018; Zellner, 2011).

The research literature comprises a considerable body of studies concerned with learning and creating music using programming (Baek and Taylor, 2020; Manaris and Kohn, 2016; Petrie, 2022). Other researchers reported on developing computational thinking among primary school students as a byproduct of music engagement (Guzdial, 1991). Research was carried out to foster the development of both coding and music skills in young students (Ludovico and Mangione, 2015; Barmpoutis, 2018). Research was also carried out to exploit the musical media to promote aspects of computational thinking among pupils of 6th to 8th grades (Barate *et al.*, 2017). The present study explores the teaching of programming through music activities. Through the pursuit of music, the need for programming knowledge will rise from the students' behalf. The music activities consist of the programming of familiar songs through which an experiential connection is created between the musical structures and the abstract programming structures, enhancing their understanding. Music activities were chosen since music also has a set of rules and structure that must be followed to play it and an abstract component in delivering the music. The creativity of music is similar to aspects of programming (hacking, algorithms, design, etc.). In addition, music frequently has a more immediate outcome than programming and frequently an effect on emotions (Koelsch, 2010; Zentner *et al.*, 2008).

2.2. *STEAM Education*

The STEM education approach was developed, integrating the Science, Technology, Engineering, and Mathematics disciplines (or part of them) to address global mission-oriented projects had to be addressed (Kim, *et al.*, 2012). The integration of the above academic disciplines was originated when projects such as brain research, sophisticated applications, finding a cure to an epidemic, and more had to be developed. Later, the 'A' was added, which stands for the arts disciplines. Among the above disciplines' integration goals is to develop skills necessary for functioning in the developing technological world (Quigley *et al.*, 2017). Liao (2016) claimed that the above disciplines could improve student engagement, creativity, and problem-solving skills and improve skills required for vocation and economic advancements, such as collaboration, communication, and adaptability. The integration creates opportunities for handling a problem from different points of view and thus nurtures creativity abilities and develops higher-order thinking and problem-solving skills.

Technological disciplines (such as programming) should be taught to students starting from an early age to prepare future citizens for the challenges of a technology-oriented world. Students develop problem-solving abilities, analytic thinking abilities, creative thinking, and computer literacy via learning programming. However, programming is perceived as a difficult task (Hava & Koyunlu Unlu, 2021) since it requires abstraction abilities that beginner students in programming do not possess yet. There should find a way to mediate these concepts via enjoyable and stimulating activities such as music that will attract their attention to be engaged in it. The MelodyCode learning unit was designed to achieve the above goal and applies the STEAM education approach by inte-

grating music, which belongs to the Art disciplines, with programming, which belongs to the Technologies disciplines.

2.3. *The Spiral Learning Approach*

The term 'spiral curriculum' was coined by Bruner (1960), who asserted that any subject could be taught in some intellectually honest form to any child at any stage of development. Meaning that even complex concepts, if presented at a level appropriate to the learner's age, can be understood by him. The main ideas underlying the spiral curriculum based on Bruner's work are: (A) Students revisit a topic, theme, or subject several times throughout their school studies; (B) The complexity of the topic or theme increases with each revisit; and (C) Revisiting a topic or theme solidifying of the learned subject matter while the student revisits it. Spiral learning refers to beginning with basic ideas and ending with complicated ones (Harden, 1999; Johnston, 2012).

Shneiderman (1977) used the spiral teaching approach to teach novice programmers claiming that this approach makes programming education more natural to students, alleviates 'computer shock,' and promotes the development of computer literacy. He (ibid) also asserted that the spiral approach is the parallel acquisition of syntactic and semantic knowledge in sequence, which increases student motivation by using meaningful examples, builds on previous knowledge, suits the student's cognitive skills, provides an understanding of recently acquired material, and develops confidence throughout the completing of increasingly complex tasks.

2.4. *Cognitive Load Theory*

Cognitive load refers to the load created on the learner's working memory resources while performing a particular learning task (Sweller, 1988). Memory resources are short-term memory (working memory) and long-term memory. When one faces new information, the working memory is limited in capacity and time. On the other hand, the capacity of long-term memory is unlimited. The information stored in it is organized in cognitive knowledge structures termed "schemas." Cognitive load can be diagnosed in two dimensions: causal and evaluation. The first represents the interaction between the task and the learner's characteristics, while the second represents the measurable characteristics of three aspects of cognitive load: extraneous, intrinsic cognitive, and actual.

Extraneous cognitive load refers to the load created by the task and the environmental requirements in which the task is performed. This load results from the interaction between the task and the learner's characteristics.

Intrinsic cognitive load refers to the amount of effort invested by the learner in performing a particular task, and therefore, this aspect represents the learner's cog-

nitive load. This effort is measured while the learner is busy performing the task or learning.

The germane cognitive load refers to schemas processing, construction, and automation of knowledge (García *et al.*, 2011). Studies have shown that cognitive load is affected by the learner's age (Gathercole *et al.*, 2004) and socioeconomic status (Siegler, and Wagner, 2005). They found an inverse relationship between economic status and age and the cognitive load created in the learning process. Measuring methods were developed to estimate cognitive load (Paas *et al.*, 2003), and one of them is based on the self-report of the learner in a retrospective reporting manner. In the present study, where students are exposed to abstract learning concepts while they are beginners at programming and, according to Neo-Piagetian cognitive theories, can not demonstrate abstract abilities to cope with programming (Morra *et al.*, 2007) (Epstein, 1980), to reduce the cognitive load, this exposure is done through the mediation of music, which provides a concrete use for the abstract programming concepts.

2.5. *Developing Programming Self-efficacy*

A positive correlation between students' beliefs regarding their academic abilities and their motivation to achieve was detected by educators (Metallidou and Vlachou, 2007). Self-efficacy is a good predictor of students' motivation and learning (Zimmerman, 2000). The concept of "self-efficacy" was coined by Bandura (1977), and the term "sense of self-efficacy" refers to the level to which the learner believes in his ability to successfully perform a task at a given condition to achieve the desired results. Bandura (1986) stated that this belief influences the behavior of the individual, her preferences, the efforts she invests in tasks, and her persistence in pursuing the goals she sets for herself. Studies showed that a sense of self-efficacy stems from the learner's perception of his knowledge, personal ability, performance, and control (Linnenbrink and Pintrich, 2010; Goddard *et al.*, 2004; Zimmerman, 2000).

For many, this belief serves as a guideline for life and a basis for action, influencing the individual's decision to behave in different situations and determining her or his ability to persist in stressful situations. People with a high sense of self-efficacy tend to respond to challenges and persevere to achieve their goals, believing that it depends solely on them and their determination. Self-efficacy has a significant impact on motivation and performance. Self-efficacy influences and is affected by the level of expertise a person demonstrates in a particular field. The higher the person's self-efficacy in the field, the more his motivation to become an expert in it increases. On the other hand, expertly in a particular field yields a sense of high self-efficacy regarding one's ability to deal with the field's challenges.

To raise students' sense of self-efficacy regarding their ability to cope with programming, and according to Neo-Piagetian theories of cognitive development to bring them to a state where they become experts in the field (Morra *et al.*, 2007), a learning environment should be developed to nurture this process.

3. The Study

Lavy's pilot study (2021) presents findings regarding the effect of using music on learning the basics of programming of seventh- and 8th-grade students. The current pilot study examines whether the above idea is also suitable for younger students. To this end, the study units have been rewritten and were modified to fit 6th-grade learning levels.

In what follows, information about the study participants, the course of the study, data resources, and analysis tools will be presented.

3.1. *The Study Participants*

Eleven 6th grade students (eight male and three female) from a regional rural elementary school participated in this study. Only four students (two male and two female) had previous music knowledge. Each of them plays a musical instrument and is proficient in reading notes. However, all the other participants or the researcher herself had no prior music knowledge. The study was conducted in weekly meetings of two lessons each at the school's computer laboratory. Participation was voluntary. The number of participants was decided according to the number of computers in the school laboratory. In April 2020, the course of the study was stopped due to the Covid19 pandemic. At that time, all schools were closed, and the students were sent home for distance learning.

3.2. *The Course of the Study*

First, it is essential to specify that although the researcher is fond of music very much, she has no musical education. Hence, to develop the MelodyCode study unit, she was assisted by a music teacher, from whom she learned the music notes so she could convert them to their representation in Scratch. It should also be noted that most of the study participants had no prior musical education. Hence, the second lesson was dedicated to teaching the students to convert music notes into their Scratch representation. The conversion was necessary since the songs' notes were taken from the website `https://www.zemereshet.co.il`, in which they appear in their formal representation. By the end of that lesson, the students could successfully program one of the songs from the above website.

The goal of MelodyCode was that via the programming of simple songs, the need for programming structures and concepts would be created by the students. Therefore, the first task in each study unit was to select a song from the above website and program it in Scratch. The level of complexity of music demands increased from unit to unit. In addition, after programming the song, the students were asked to develop an animation or game that would suit the spirit of the song. During the students' engagement with the song programming, a need for programming constructs was created. These needs have

discoursed during the class discussions, and the researcher introduced the relevant programming structures and concepts to address these needs. For example, when students were tackled with the problem of finding an incorrect note among a long list of notes, they suggested dividing the whole list of music notes into groups. At this point, the concept of function (Block in Scratch terminology) was presented. The next raised question was, what would be the logical division into blocks of the whole list. They suggested that each line of musical notes be converted into one block.

In the design phase of MelodyCode, a match between music elements and corresponding programming constructs and concepts was performed (Table 1). As was previously mentioned, the music involved in the MelodyCode concerned popular songs. The popular songs selected were Jewish holidays songs. The reasons for choosing music related to Jewish holidays are related to the study participants' age and the desire to connect the learning process to traditional and cultural experiences. The Jewish holidays have memorable songs, foods, and customs and have an essential role in the students' social and traditional life, and as such, have an integral part in the elementary school curriculum in our country. Therefore, it seems appropriate to focus the learning of programming fundamentals around a central theme that deals with the music of the Jewish holidays that have been familiar to students since the dawn of their childhood. Via the programming of the songs, the need for programming constructs and concepts such as variables, functions, repetitive commands, parameters, parallel processes, and more was raised. During two Jewish holidays, the students were engaged in developing a project that included programming one of the near holiday songs accompanied by a suitable animation or a game to fit the chosen song. Nearby the holiday date, a competition that win awards was held to rate the best three projects according to the pre-selected criteria list. The students decided on the criteria list and the relative weight adjusted to each criterion in a class discussion devoted to this purpose. The competitions allowed the students to demonstrate their abilities and significantly contributed to shaping students 'social status. The research literature indicates a positive effect of academic competition on the learning process (Burguillo, 2010; Fülöp *et al.*, 2007; Sheridan and Williams, 2011). It was found that friendly competitions enhance students'

Table 1
Mapping corresponding elements of music and programming concepts

Music	Programming
Playing a specific note for a specific duration	Using the concept of variable
Writing song notes according to its verses	Using the concept of function
Replaying of the same music segment	Using the concept of a simple loop
Playing the same music with various instruments	Using the concept of objects and parallel processes
Playing the same music segment in different octaves	Using the concept of variable and a parameter and parallel processes
Polyphonic music	Parallel processes
Playing a music canon	Using the 'broadcast' block command
Writing music chords and merging them into a melody	Structured programming
Adding the accompaniment of a drum to a song	Nesting loops

motivation to learn and help in increasing their learning performances. Fülöp (2004) defined constructive competition as a social and cultural phenomenon that enhances children's abilities, develops their ambitions, and motivates their learning.

The best three projects won small prizes representing the holiday spirit, while the rest of the students received consolation prizes to avoid disappointment. The projects' requirements became more complex from one holiday to the next. The last project was carried out in teams to enable the students to experience teamwork. This project was interrupted near its end due to the Covid19 pandemic.

The Scratch environment was presented to the participants in the first lesson focusing on the music blocks. Each block command was explained, followed by a demonstration of its use.

Table 2

Schematic description of the contents of the weekly lessons

Lessons	Jewish holiday	Music	Programming concepts	Tasks	The average level of difficulty (1–9)	
					Programmed melody	Animation/ game
1–4	Rosh Hashanah[1] – Jewish new year	Notes, duration playing of a note, tempo, music segment, music instruments	The Scratch block of a music note, function, simple loop	Writing the script of one of "Rosh Hashanah" songs from: https://www.zemereshet.co.il and design a simple animation to fit the melody	5.1	6
6–10	Hanukkah[2]	Playing the melody in different octaves	Functions, loops, variables,parallel processes	Writing a script of one of "Hanukkah" songs and playing it in two different octaves in concert, and designing a suitable animation or a simple game	4	5.5
11–14	Tu BiShvat[3]	Playing a melody with two hands	Functions, loops, variables Parallel processes	Writing a script of "Tu BiShvat" song played with two hands in concert and designing a suitable game for two players	4.2	5.8
16–19	Purim[4]	Playing Song String and Adding drums accompaniment	Use the "launch" command, nesting loops	Writing two scripts of two different "Purim" songs, played in concert, adding drums accompaniment, and designing a suitable animation or a game	4.1	5.6

[1] "Rosh Hashanah" meaning "head of the year", is the Jewish New Year.

[2] "Hanukkah" is a Jewish festival commemorating the recovery of Jerusalem and subsequent rededication of the Second Temple at the beginning of the Maccabean revolt against the Seleucid Empire in the 2nd century BCE.

[3] "Tu BiShvat" is a Jewish holiday occurring on the 15th day of the Hebrew month of Shevat. It is also called the 'New Year of the Trees'. In contemporary Israel, the day is celebrated as an ecological awareness day, and trees are planted in celebration.

[4] "Purim" is a Jewish holiday which commemorates the saving of the Jewish people from Haman, an Achaemenid Persian Empire official who was planning to kill all the Jews at the 5th century BCE.

The students were handed a study unit that included several exploration tasks from the second meeting and on. Each study unit contained four to eight exploration tasks with an increasing level of complexity. The MelodyCode is designed according to spiral learning, meaning that concepts and programming structures were revisited several times at an increased complexity level. The students worked individually on the exploration tasks and could turn to the researcher when he/she tackled difficulties. The last fifteen minutes of each session were dedicated to a class discussion in which the newly learned concepts were summarized, and insights gained during the activities were shared. In addition, the students were asked to provide feedback on the study units, refer to the clearness/ comprehension, and whether extra explanations were needed. It was made clear to them that they could express their opinion freely. Table 2 shows a schematic description of the weekly lessons, including the average value of their self-reported level of difficulty they experienced in each project (programmed melodies, animation/game development).

3.3. *Research Aim and Derived Questions*

This pilot study aims to explore the effectiveness of MelodyCode for learning programming fundamentals among 6th-grade students. The derived research questions are:

1. What were the students' preconceptions regarding programming, and how were they influenced (if any) by the learning via MelodyCode?
2. How does learning via MelodyCode influence students' self-concept regarding their ability to cope with programming?

3.4. *Data Resources and Analysis Tools*

The study is exploratory, and it is part of broader research that explores the effectiveness of developed study units for teaching mid-school and elementary school students how to program via music. The data resources of the present study were:

a) **In-depth semi-structured pre and post-interviews:** to follow the change in perceptions regarding programming, pre and post semi-structured interviews were conducted with all the study participants. In the pre-interviews that were conducted at the beginning of the course, there were four leading questions: (1) what do you think about programming; (2) what are the required skills to be able to be a good programmer; (3) why did you choose to join the course; (4) express your opinion regarding the following: I think I can succeed in getting mastery of programming.

They conducted the post-interviews to reveal the qualitatively different ways the participants experienced, perceived, and understood the learning to program via music. The leading questions in the post-interviews were based on Ornek's study (2008) and were modified to suit the present study. In addition, they were asked to rank the level of difficulty they had experienced after each project,

referring separately to the music programming and the animation/game development. The ranking varied from 1 to 9, while 1 indicates very low cognitive load, and 9 indicates a very high cognitive load. They were also asked to justify their ranking. The self-esteem index questionnaire is based on Paas *et al.* (2003) to estimate the cognitive load students experience while coping with different tasks. Due to the Covid-19 pandemic, the interviews were conducted using the Zoom platform.

b) **The students' feedback:** The students were asked to provide feedback regarding the study units, including positive and negative reviews, and email it to the researcher. The students' feedback is intended to improve the following study units.
c) **The student outcomes:** Tracing the students' design and development of their projects could shed light on their curiosity, enjoyment, and motivation to learn.
d) **The researcher's reflective journal:** After each lesson, the researcher documented episodes during the meetings and the insights she gained.
e) **Transcripts of informal talks conducted with the Computer Lab Director.** She was present in most lessons to provide technical problems. She provided an additional point of view on the learning process.

The present research was designed to suit action research (Stringer, 2013), which aims to learn from the feedback received from the study participants and from the researcher's insights to improve the following study units (Denscombe, 2014). Action research includes the following stages: (1) detect a change; (2) examine the present situation; (3) design different interventions; (4) apply the intervention; (5) examine the impacts of the intervention; (6) assess against initial goals; and (7) distribute findings.

In the current study, the understanding that traditional methods used to teach programming deter students (predominantly female ones) from choosing programming studies has led to the development of a different learning approach that integrates two disciplines: music and programming (addressing stages 1–3). In a weekly two lessons session, the students were engaged with a MelodyCode learning unit. (stage 4). In each session, participant observation was conducted by the researcher and was documented in detail in the researcher's reflective journal. The insights gained by the researcher and the students' feedback were analyzed and evaluated (stages 5 and 6). The insights gained from the previous stage were implemented in the following study unit (stage 7).

The Phenomenography method (Marton, 1986) was the most suitable for the data analysis. This method suits educational research that aims to discover the qualitatively different ways students experience, perceive, and understand various aspects of a new phenomenon (Bowden *et al.*, 1992) that, in the present study, refers to learning how to program via music. According to Sjöström and Dahlgren (2002), phonomyography analysis includes the following steps: (1) familiarization of the researcher with the research data. It is done using reading through the transcripts of the gathered data; (2) compilation of responses from participants to a particular question. The researcher should identify the most significant elements in responses given by participants; (3) a

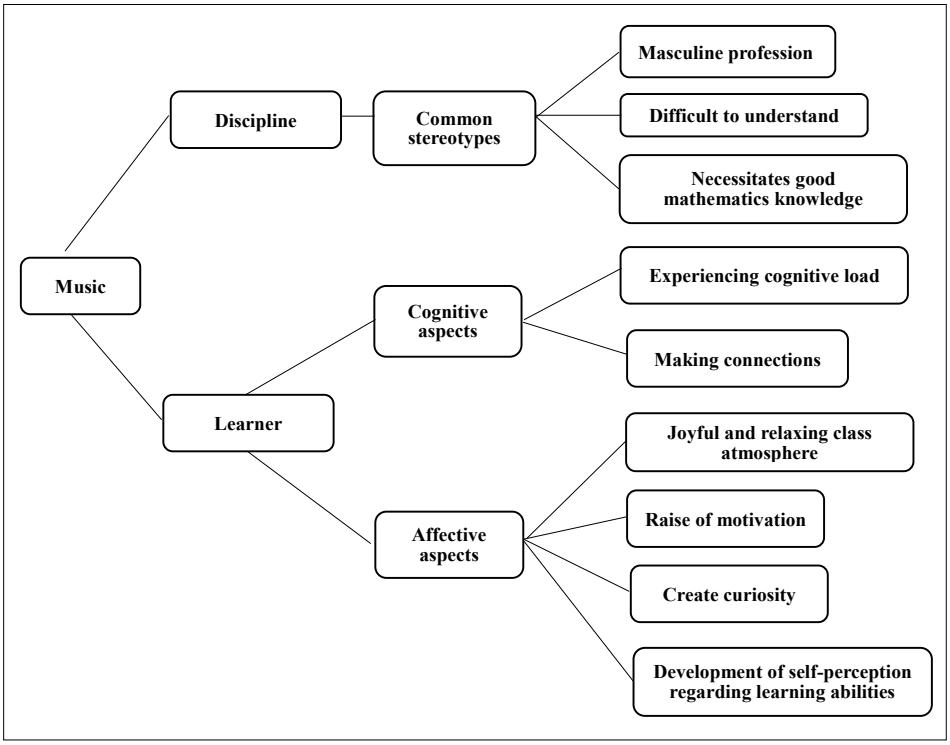

Fig. 1. Schematic description of data organization.

preliminary grouping or classification of similar responses; (4) a preliminary comparison of categories; (5) naming of categories; and (6) a contrastive comparison of categories. It includes the categories' descriptions and similarities between them.

Analysis of the data revealed references to two main issues – views related to programming and views related to the learner's self-ability to cope with learning programming and how the music affected these views (Fig. 1). In the following section, elaboration on these issues will be presented and discussed.

3.5. *Ethical Issues*

Since the researcher and the study participants had an authority relationship (teacher-student), prior to the study conduction, the participants were informed about the researcher's intention to conduct a study that would follow their experiences using the data from their interviews and learning products to receive their consent to the process. The study received the school principal and the students' parents' approval. The students were free to choose whether to participate in the study. All the study participants expressed their consent to take part in the study. Pseudonyms were used for all the study participants to keep their privacy and confidentiality.

4. Results and Discussion

As was previously mentioned, analysis of the data revealed references to two main issues – views related to programming as a discipline and views related to the learner's prior self-perception regarding her or his ability to cope with learning programming and how MelodyCode affected these views. Before elaborating on the results, It should be specified that all the presented results should be treated with caution because of the small sample size.

As to programming, students referred to common stereotypes such as a complex topic to understand; being a good programmer necessitates good mathematics knowledge and a masculine profession. From the learner's point of view, there are references to cognitive and emotional aspects (Fig. 1). The quotes of the students presented below were translated from Hebrew. However, an effort was made to present a translation that expresses their ideas most reliably.

4.1. *Students' Preconceptions Regarding Programming*

Elementary school students would have preconceptions about programming even if they did not learn it yet. These preconceptions result from socialization and their own experiences (if any) (Geldreich *et al.*, 2019).

To address the first part of the research question, and as was stated earlier, in a pre-interview conducted with all the study participants, they were asked to relate to the following questions: (1) what do you think about programming; (2) what are the required skills to be able to be a good programmer; (3) why did you choose to join the course; (4) express your opinion regarding the following: I think I can succeed in mastering programming.

Analysis of the pre-interviews revealed the following: Regarding the first question, all the students said in similar words: "*We did not learn to program yet, but my brother, who is in eleventh grade, told me it is a complicated subject.*"; "*My dad works in high-tech, and he told me that I should strive to study how to program because it is a lucrative profession.*"; "*In this subject [programming], one can learn to communicate with the computer and create interesting games, similar to the ones that we play with.*"; "*On TV, it is constantly said that this profession is in great demand and lacks many workers, especially girls.*"

Students' preconceptions regarding programming are created from the information they receive at home from parents, relatives, and the media (Geldreich *et al.*, 2019). As a result, these preconceptions affect their learning preferences on whether to choose programming studies or not. Kong *et al.* (2018) referred to the phenomenon that elementary school boys are more interested in computer programming than girls and attributed it to widespread media stereotypes. Boys' interest in programming came to fruition in their relatively high percentage of study participants (eight out of 11 students). The widespread media stereotype of male computer specialists serves as role models for boys. However, it negatively influences girls when other considerations are involved (Master

et al., 2016; Ensmenger, 2010). In recent years, the above situation has changed, and a higher percentage of women can be found in programming professions, especially when they start this career before starting a family (Buhnova and Prikrylova, 2019). On the other hand, there is a tension between these preconceptions and the social message that programming is a lucrative profession (Ensmenger, 2010); hence one should invest efforts to become a programmer.

As to the second question, all the students said in similar words: "*To be a good programmer, you have to be good at math and know English as well.*"; "*You need to be smart and ready to spend many hours solving problems*." Similar results were found among seventh and eighth-grade students (Lavy, 2021). Attributed traits to good programmers are derived from the traits of famous key figures in computer science (Master *et al.*, 2016).

As to the third question, male students' answers in similar wording were: "*I enjoy playing computer games. I want to learn how to design games by myself*". Female students' answers in similar wording were: "*You told us that the learning includes music; this raised my curiosity because I like music very much.*" "*The only reason I joined the course was the music. I hope it will be interesting*". In the answers to this question, one can discern the difference between boys' and girls' preconceptions regarding programming. While the boys expressed their wish to develop computer games independently, the girls come from an apologetic stance saying they joined the course because of the music and not necessarily because of the programming and expressed a wish that the lessons would be engaging. Similar results were found by Master *et al.* (2016).

As to the fourth question, male students said in similar wording: "*I managed to develop simple computer games before. I want to improve these games and raise their complexity*"; "*I want to be a developer, but I have some concerns because I am not that good in mathematics*". On the other hand, female students said in similar wording the following: "*I hope that the music will ease the learning of the programming.*";"*I want to believe I can, but I am not sure*." Similarly to the third question, the students' reactions differ according to their gender. While most male students demonstrated self-confidence in their ability to cope with programming, a minority expressed concerns about encountering difficulties since they are not very good at mathematics. On the other hand, the female students expressed low self-confidence regarding their ability to handle programming (Ensmenger, 2010), hoping the music would make a difference. Although the above results reflect the study participants' views, they should be referred to cautiously due to the small sample of the research group.

4.2. *The Effect of MelodyCode on Common Stereotypes*

In what follows, results regarding the second part of the first research question are presented and discussed. During the post-interviews conducted via the Zoom platform, most of the students referred to the effect of MelodyCode on their views regarding programming that stemmed from common stereotypes.

David: *"Most computer games involve battles that usually attract boys while girls are reluctant to play these games. As a result, the programming profession is*

perceived as a male profession, and most of the girls I know do not play these games. The need to develop an animation or computer game that a certain holiday song will accompany raised the need to think of other options besides characters fighting on screen."

Noga: *"Learning programming is difficult because it is different from spoken language in which one can understand your idea even if you expressed it implicitly. Here, if you don't write the correct command and refer to all possible cases, your solution is not complete and correct. Also, to be able to write a correct computer program, you need to think of the relevant concepts that fit the task needs but were unknown to us at this time. The engagement in music programming allowed us to practice it in an attractive way."*

Jonathan: *"I used to think that you must be very clever and be good in mathematics to become a good programmer. My views were based on information I absorbed at home. MelodyCode presented us with other alternatives besides programming solutions to mathematical problems. You can learn programming through music engagement. Tell it to the potential students for the next course; many of my classmates wanted to join the course but were afraid that they have to be very good at mathematics as well."*

Before relating to David's excerpt, it should be noted that he had no prior music education before joining the course. We can learn one of the underlying reasons programming is considered a masculine profession. Many boys have been more attracted to battle computer games from childhood than girls. Following the evolution of programming reveals that this profession requires many working hours. It is perceived as a male profession (Ensmenger, 2010) since, in most cases, women take responsibility for taking care of the family children cannot afford the investment of many hours to develop and advance in this profession (Ensmenger, 2015).

In her excerpt, Noga refers to the difference between the spoken language and programming in that the programmed solution must be both accurate and comprehensive in the sense that it must address all the possibilities embedded in the problem. To be able to do so, one should demonstrate analytical thinking skills and abstraction skills (Tsalapatas *et al.*, 2011; Kramer, 2007; Bennedsen and Caspersen, 2006). Moreover, being beginners in programming, the study participants are expected to demonstrate abilities that do not fit Neo-Piagetian cognitive theories(Morra *et al.*, 2007). According to these theories, people can demonstrate abstract abilities only when they become experts in a particular field.

Noga refers to music engagement's contribution to coping with the situation described above, saying that through the music programming, the need for the programming concepts emerged, which eventually helped her understand these concepts (White and Sivitanides, 2002).

As to Jonathan's excerpt, research showed that students' mathematics ability was strongly related to their programming performance (Qian and Lehman, 2016). However, no indication regarding the potential of being a good programmer with average mathematical ability was detected (Pianta *et al.*, 2008). In the above excerpts, Jonathan de-

scribes how MelodyCode changed his views regarding programming saying that learning programming through music encourages the student to discover various ideas for games and animations that do not necessarily require mastery of mathematical knowledge. Finally, Jonathan urged the researcher to share the change in his preconception regarding the need to be good at mathematics with other students who think the same and, as a result, avoid learning to program.

4.3. *The Effect of MelodyCode on Cognitive Aspects*

The study participants were asked to rank the degree of difficulty they experienced while working on the projects with separate reference to music programming and application/game development (Table 2). The degree of difficulty can serve as an indicator of the cognitive load they experience (Paas *et al.*, 2003; Paas *et al.*, 1994). They were also asked to justify their ranking. The data obtained show that the students' self-reported level of difficulty regarding the music programming was lower than the one regarding the development of application/game in all the projects. The justifications attached to the ranking relating to the music programming related mainly to affective issues such as "The fun pursuit of music made up for the difficulty, so overall I was not left with an experience of great difficulty." However, the justifications attached to the programming of the animation/game referred mainly to cognitive aspects. In what follows, elaboration on cognitive aspects is presented and discussed.

Yoav: *"I hesitated a lot before I joined the lessons because I was sure that I would not be able to cope with programming. My concerns stemmed from rumors that programming is difficult to learn and understand. When I successfully programmed a song's melody, these concerns faded from the first lesson. As if the focus of learning moved to the joyful part – the music while the programming became the tool to achieve it."*

Interview segment:

Researcher: *"Could you please describe your working process on a project?"*

Mika: *"The assignments regarding the music programming in the MelodyCode units enabled a sense of success right after you finished the task. You could hear the success since it meant that your programmed melody was correct. This success motivated me to pursue the development of a suitable animation or game."*

Researcher: *"Can you give an illustration from the course?"*

Mika: *"Yes. While working on the Rosh Hashanah project, we selected a song and programmed its script. After running each stanza of the song, it was possible to determine if the notes were correct or not. Once the song was ready, I began to think of an animation that would suit it and reflect the holiday spirit. Dur-*

ing writing the music script, a need to use the concept of function [block in Scratch] emerged. This concept was unfamiliar to me before. Linking between the programming concept and its musical concrete use helped me understand and remember it."

Researcher: *"What programming concept reminds you of drum accompaniment?"*

Mika: *"It is easy. Nesting loops."*

Researcher: *"What programming concepts remind you of playing in different octaves?"*

Mika: *"Variables and parameters."*

Among other students who expressed the same ideas in similar words, Mika referred to the assignments' structure they had to cope with, which was modular and enabled focusing on one mission at a time and receiving immediate feedback on their performances. Beginning first with the task of music programming provided students with immediate feedback (correct melody) and created motivation to learn programming structures that would help them advance in the task. In addition to being a fun pursuit, the music programming engagement revealed a concrete use of abstract programming structures. Driven by their success in the music task, the students successfully developed an animation/game that would suit the music they had created. When developing an animation/game, they were already familiar with the required programming structures. Nevertheless, they reported experiencing a greater difficulty than they had experienced in the music task since they had to cope with developing an animation or a game that suits the holiday spirit, which requires creative abilities. According to the research literature, the cognitive load that might develop while engaging in complex topics such as programming is higher among beginner students than in novice ones (Gathercole *et al.*, 2004).

In terms of cognitive load theory (Sweller, 1988; Sweller *et al.*, 1998), we might say that according to Yoav and Mika, learning via MelodyCode enabled the students' coping with the intrinsic cognitive load created during the learning process. The intrinsic cognitive load refers to the cognitive efforts associated with a specific topic, which in our case, is the learning of abstract programming concepts and structures that are not familiar to them. The exposure to the abstract programming concepts via music programming engagement enabled the students successfully cope with the abstract concepts (Epstein, 1980).

The germane cognitive load refers to processes in working memory, leading to the construction and automation of cognitive knowledge structures (schemas) (Sweller, 1988; Sweller *et al.*, 1998; García *et al.*, 2011). These processes do not happen automatically but depend on the learner's motivation (Sweller *et al.*, 1998). Via the learning using the MelodyCode, links were created between exciting activities such as drum accompaniment and abstract concepts such as nesting loops. These links might initiate the construction of knowledge schemas that would develop in future engagements with these abstract concepts.

4.4. *The Effect of MelodyCode on Emotional Aspects*

Data and discussion regarding the second research question are presented in what follows. In the interviews, the students referred to the following emotional aspects: joyful and relaxing class atmosphere, raising motivation, creating curiosity, developing self-perception regarding learning abilities. Herein are representative excerpts from the students' interviews:

Yuval: *"When you [the researcher] came to our class and explained to us how we would learn to program through music, I imagined joyful lessons in the sense that music will be heard from all computers. I must admit that my hopes were fulfilled".*

Mika: *"The lessons were totally different from the lessons of other subjects we learn. At what lesson can one hear her programmed music? We did not feel that we were coping with a difficult topic. If we were learning only the programming, I'm not sure I would feel the same way. The music made the difference and changed everything."*

Gal: *"I felt thrilled when the song's melody, whose notes I programmed, sounded correct and without mistakes. This joy spurred me to invest efforts in a computer game that would suit the music. It was important to me to develop a game that attracts classmates' desire to play."*

Ziv: *"As I was writing the code for the Rosh Hashanah song, I had questions like How to find incorrect musical notes in the script easily? How to avoid repeating the same segment of commands? How to easily change the melody octave? How to program polyphonic music? I was so enthusiastic that I wanted to know the answers to all of the above questions right from the beginning."*

David: *"Even though each student worked individually on his tasks, you could not avoid hearing other classmates' melodies. This raised my curiosity regarding computing commands used by other classmates, so I turned to ask for their help to improve mine."*

Yuval: *"During the lessons, we felt free to move around and learn from other classmates' performances. Sometimes students helped me and sometimes I helped them, and it allowed everyone to improve their projects."*

Jenkins (2002) said:

> *"Learning (or perhaps here 'being taught') programming can be very dull. Lectures covering syntax details are never going to be especially inspiring, and exercises that involve simple mathematical manipulations of collections of student marks, stock levels, baseball statistics, or bank account details are never going to set the pulse racing. Yet a glance in many programming texts will yield many turgid examples of each of these."* (p. 56)

As to experiencing feelings of enjoyment and interest, he (Jenkins, 2002) added:

> *"At its best programming can be an enjoyable, creative activity, and many students derive great enjoyment from their programming. They enjoy it even more (and learn more) when they can work on assignments that inspire them. It is a shame that so few assignments do indeed inspire"* (p. 56)

From the above excerpts and the evidence from the researcher's reflective journal, it might be said that learning programming via music made it happen.

From Yuval's excerpt, we can learn that his expectations from the course are not typical. In general, students' expectations of a new learning method refer to cognitive aspects such as facilitating the learning process and enabling them to acquire new knowledge and understand the material being taught rather than encounter difficulties (Jin and Hill, 2001; Littlejohn *et al.*, 2010). In the case of the learning using the MelodyCode, emotional expectations were also raised. The expectation of enjoying the learning process stems from integrating music into the learning process since music serves as one of the primary sources of enjoyment (Batt-Rawden and DeNora, 2005). When MelodyCode was introduced to the students, they expected the music to change the usual learning routine.

Referring to Mika's and Gal's excerpts, a reference to creating two kinds of the joy of learning, active and passive, is detected. The passive joy of learning means contentment with a pleasant state. The learner's immediate reaction is issued by an external factor beyond one's control. The active joy of learning is a state that results from the students' effort. The emotion of achievement results from sudden and surprising success (Varila and Viholainen, 2000; Rantala and Maatta, 2012). Learning how to program via music nurtured both kinds of joy. MelodyCode was designed as a collection of engaging activities, which added to the passive joy, while the students' success in creating animation or a computer game to accompany their programmed music added to the active joy. It is also in line with research findings that using untraditional approaches in learning and incorporating engaging activities such as games positively affect the class atmosphere (Yu, 2005).

Referring to Ziv's excerpt, he describes the effect of music engagement on his curiosity to pursue the learning of programming structures and concepts to address the musical issues he tackled. Curiosity can be articulated as the tendency to inquire, explore, or pursue knowledge. It is simply the frame of mind in which one wants to learn more about something. Curiosity can be viewed as a source of internal motivation that encompasses the foundation of education (Binson, 2009). The motivational fuel for learning at each step of the educational process is driven mainly by curiosity. Curiosity motivates students to learn more and more about their world, and as a result, a deeper understanding of the interactions and the relationship between the various elements is achieved (Loewenstein, 1994; Binson, 2009).

To illustrate students' eagerness to learn new programming concepts. Students were asked to create a drum accompaniment to one of the songs they had programmed. They

had to begin with performing an accompaniment of only two drums. The researcher demonstrated drum accompaniment using the concept of nesting loops. Right after that, they turned to the computers to practice it. Some (the students with prior music knowledge) asserted that different songs have different rhythms, requiring different drum accompaniment rhythms. The enthusiasm was high among all students, which yielded programmed drum accompaniments to all the songs they had programmed so far without being asked to do so.

As to the development of self-perception regarding learning abilities, representative excerpts are presented:

Noga: *"I'm not very good at mathematics, so I had concerns that I will encounter difficulties. I joined the course because of the music and stayed because of the programming. I belong to the music class at school and hoped that my music specialty would help me cope with programming. When I finished codding the melody successfully, it raised my self-confidence that I would also be able to overcome the programming of the animation. After finishing the animation to my satisfaction, I was proud of myself, even more, when classmates asked for my help."*

Ziv: *"My success in programming the song gave me a good feeling regarding my ability to cope with the following task. This feeling influenced my willingness to cope with the programming of the animation. If it was the other way around, I am not sure I would have acted the same."*

From the above quotes, it can be concluded that learning via MelodyCode played an essential role in affecting the students' self-perception regarding their ability to cope with programming tasks. The student's engagement in programming a familiar song provided positive feedback that helped increase their self-perception regarding their ability to cope with the following task involving programming an animation/computer game. The task design in MelodyCode was meant to build their self-confidence in coping with programming gradually. Success and a sense of self-efficacy are interrelated (Bandura, 1977). Success increases a sense of self-efficacy, and a high sense of self-efficacy affects the behavior of the individual, her motivation, her choices, the efforts she invests, her ability to cope with different situations, and her perseverance in pursuing the goals she sets for herself (Bandura, 1986). Hence, we may conclude that learning programming fundamentals via MelodyCode provided the students with success opportunities, increasing their self-efficacy to cope with programming.

4.6. *Concluding Remarks*

As in the Pied Piper legend where the music had its magic influence on the city rats and then on the children of Hamelin to follow the flutist outside of the city, the music role in the MelodyCode study unit is to raise students' motivation and curiosity to be acquainted with programming structures and concepts. The music induced a relaxed class atmosphere and provided a kind of "softened covering" to programming, which

is considered a complex and challenging topic to cope with (Bosse and Gerosa, 2017; Derus and Ali, 2012).

Data analysis revealed that learning via MelodyCode changed the students' stereotypes regarding programming. Stereotypes such as programming is a male profession requiring mathematics excellence and more. The data analysis also revealed that learning using MelodyCode affected both the students' cognitive and emotional perspectives.

As to cognitive perspectives, the students were exposed to attractive situations that evoked the use of programming structures and concepts, raising their motivation to understand and use them. Making connections between music structures and programming structures also affected the cognitive load (Sweller, 1988) the students experienced during their engagement in the tasks.

As to emotional perspectives, references to a pleasant class atmosphere during lessons, raising motivation and curiosity, and increasing self-perception in their ability to cope with the difficulty of learning how to program was detected. There is a reciprocal relation between self-efficacy and motivation. High sense of self-efficacy in one's ability results in an increase in her or his motivation and vice versa (Bandura, 1986). The learning via MelodyCode provided the students with opportunities to experience success (correct melody of the music programming) which raised their sense of self-efficacy and motivated them to cope with the programming of animations or games.

Analysis of the transcripts of informal talks conducted with the Computer Lab Director revealed that some students experienced a positive change in their social status in the class. From being socially isolated to a state where all students seek their friendship, asking for their advice and help. There is a well-known problematic phenomenon in which some students experience social isolation in school, and many efforts are invested in minimizing this phenomenon (London and Ingram, 2018). In the first project competition, a student with problematic social status won. After winning, all students sought his company, and the positive change in his social status was observable. The competitions allowed the students to present their abilities and, hence, made a significant contribution to shaping students' social status. Studies examining the relationship between classroom social status and academic achievement have shown that low social status is associated with low achievement (Destin *et al.*, 2012; Van Laar and Sidanius, 2001). They found that students perceived to be at higher social status levels were among the high scores achievers in class. In our case, we notice a situation where academic success improves students' social status in the classroom. The change in their social status in class has also contributed to raising their motivation to develop exciting games that will be perceived as popular among classmates, and the number of classmates playing with those games will increase.

As mentioned earlier, the researcher had no prior music knowledge, and assistance from a music teacher enabled the development of the MelodyCode units. This fruitful collaboration is at the heart of the STEAM education approach and the multi-disciplinarity approach, particularly encouraging cooperation between different domains. In addition to gaining programming knowledge, during the students' engagement with the

MelodyCode units, they also become acquainted with music education, such as octaves, chords, drum accompaniment, polyphonic music, and more.

As was previously mentioned, neither the teacher nor the students need to possess prior music knowledge to benefit from the learning via MelodyCode. Fruitful collaboration with music teachers might yield additional ideas to incorporate into the study units. Hence, the course will be repeated in the following academic year with several classes of six-grade students after upgrading the study units. In addition to exposing 6th-grade students to programming through music, the learning of programming in the following learning years should be continued while adjusting the level of learning to the student's cognitive abilities. Efforts should be made to Increase the students' interest in programming studies by incorporating activities like dancing and/or reality topics popular among students at these ages.

References

Baek, Y., Taylor, K. (2020). Not just composing, but programming music in group robotics. *Music Education Research*, 22(3), 315–330.

Bamberger, J. (1982). Revisiting children's drawings of simple rhythms: A function for reflection-in-action. In: S. Strauss (Ed.), *U-shaped Behavioral Growth* (pp. 191–226). New York, NY: Academic Press.

Bamberger, J. (2000). *Developing Musical Intuitions*. New York, NY: Oxford University Press

Bandura, A. (1977). Self-efficacy: toward a unifying theory of behavioral change. *Psychological Review*, 84(2), 191.

Bandura, A. (1986). *Social Foundations of Thought and Action: A Social Cognitive Theory*. Englewood Cliffs, NJ: Prentice-Hall.

Barate, A., Formica, A., Ludovico, L.A., Malchiodi, D. (2017). Fostering Computational Thinking in Secondary School through Music.

Barmpoutis, A. (2018, March). Integrating algebra, geometry, music, 3D art, and technology using emoticoding. In 2018 *IEEE Integrated STEM Education Conference* (ISEC) (pp. 30–33). IEEE

Bell, J., Bell, T. (2018). Integrating Computational Thinking with a Music Education Context. *Informatics in Education – An International Journal*, 17(2), 151–166.

Batt-Rawden, K., DeNora, T. (2005). Music and informal learning in everyday life, Music *Education Research*, 7(3), 289–304.

Bennedsen, J., Caspersen, M.E. (2006). Abstraction ability as an indicator of success for learning object-oriented programming? *ACM Sigcse Bulletin*, 38(2), 39–43.

Binson, B. (2009). Curiosity-based Learning (CBL) Program, *Online Submission*, 6(12), 13–22.

Bitner, B.L. (1991). Formal operational reasoning modes: Predictors of critical thinking abilities and grades assigned by teachers in science and mathematics for students in grades nine through twelve. *Journal of Research in Science Teaching*, 28(3), 265–274.

Bloom, B.S. (1956). *A Taxonomy of Educational Objectives*. New York, NY: Longmans.

Bosse, Y., Gerosa, M.A. (2017). Why is programming so difficult to learn? Patterns of Difficulties Related to Programming Learning Mid-Stage. *ACM SIGSOFT Software Engineering Notes*, 41(6), 1–6.

Bowden, J., Dall'Alba, G., Martin, E., Laurillard, D., Marton, F., Masters, G., ... Walsh, E. (1992). Displacement, velocity, and frames of reference: Phenomenographic studies of students' understanding and some implications for teaching and assessment. *American Journal of Physics*, 60(3), 262–269.

Bruner, J.S. (1960). *The Process of Education*. Cambridge MA, Harvard University Press.

Buhnova, B., Prikrylova, D. (2019, May). Women want to learn tech: Lessons from the Czechitas education project. *In 2019 IEEE/ACM 2nd International Workshop on Gender Equality in Software Engineering* (GE) (pp. 25–28). IEEE.

Burguillo, J.C. (2010). Using game theory and competition-based learning to stimulate student motivation and performance. *Computers & Education*, 55(2), 566–575.

Denscombe, M. (2014). *The Good Research Guide: For Small-scale Social Research Projects*. McGraw-Hill Education (UK).

Derus, S.R.M., Ali, A.Z.M. (2012). Difficulties in learning programming: Views of students. In *1st International Conference on Current Issues in Education* (ICCIE 2012) (pp. 74–79).

Destin, M., Richman, S., Varner, F., Mandara, J. (2012). "Feeling" hierarchy: The pathway from subjective social status to achievement. *Journal of Adolescence*, 35(6), 1571–1579.

Ensmenger, N. (2010). Making programming masculine. *Gender Codes: Why Women are Leaving Computing*, 115–141.

Ensmenger, N. (2015). Beards, Sandals, and Other Signs of Rugged Individualism: Masculine Culture within the Computing Professions. *Osiris*, 30(1), Scientific Masculinities (January 2015), 38–65.

Epstein, H. (1980). Some biological bases of cognitive development. *Bulletin of the Orton Society*, 30, 46–52.

Epstein, H. (1990). Stages in human mental growth. *Journal of Educational Psychology*, 82, 876–880.

Fülöp, M. (2004). Competition as a culturally constructed concept. In C. Baillie, E. Dunn, Y. Zheng (Eds.), *Travelling Facts: The Social Construction, Distribution, and Accumulation of Knowledge* (pp. 124–128). Frankfurt/New York: Campus Verlag.

Fülöp, M., Ross, A., Pergar Kuscer, M., Razdevsek Pucko, C. (2007). Competition and cooperation in schools: An English, Hungarian and Slovenian comparison. In F. Salili & R. Hoosain (Eds.), *Culture, Motivation, and Learning: A Multicultural Perspective* (pp. 235–284). Charlotte, NC: IAP.

García, M., Llinares, S., Sánchez-Matamoros, G. (2011). Characterizing thematized derivative schema by the underlying emergent structures. *International Journal of Science and Mathematics Education*, 9(5), 1023–1045.

Gathercole, S.E., Pickering, S.J., Ambridge, B., Wearing, H. (2004). The Structure of Working Memory From 4 to 15 Years of Age. *Developmental Psychology*. 40(2), 177–190.

Goddard, R.D., Hoy, W.K., Woolfolk-Hoy, A. (2004). Collective efficacy: Theoretical developments, empirical evidence, and future directions. *Educational Researcher*, 33, 3–13.

Geldreich, K., Simon, A., Starke, E. (2019, October). Which Perceptions Do Primary School Children Have about Programming?. In *Proceedings of the 14th Workshop in Primary and Secondary Computing Education* (pp. 1–7).

Gold, B.P., Frank, M.J., Bogert, B., Brattico, E. (2013). Pleasurable music affects reinforcement learning according to the listener. *Frontiers in Psychology*, 4, 541.

Guzdial, M. (1991). *Teaching Programming with Music: An Approach to Teaching Young Students about Logo*. Logo Foundation.

Harden, R.M. (1999). What is a spiral curriculum? *Medical Teacher*, 21(2), 141–143.

Jenkins, T. (2002). On the difficulty of learning to program. In Proceedings of *The 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, pp. 53–58.

Jin, L., Hill, H. (2001). Students' expectations of learning key skills and knowledge. *International Journal of Language & Communication Disorders*, 36(1), 333–338.

Johnson, D.C. (2011). The effect of critical thinking instruction on verbal descriptions of Music. *Journal of Research in Music Education*, 59(3), 257–272.

Johnston, H. (2012). *The Spiral Curriculum. Research into Practice*. Education Partnerships, Inc.

Kalelioğlu, F., Gülbahar, Y. (2014). The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from the Learners' Perspective. *Informatics in Education*, 13(1), 33–50.

Kim, S.W., Chung, Y.L., Woo, A.J., Lee, H.J. (2012). Development of a theoretical model for STEAM education. *Journal of the Korean Association for Science Education*, 32(2), 388–401.

Koelsch, S. (2010). Towards a neural basis of music-evoked emotions. *Trends in Cognitive Sciences*, 14(3), 131–137.

Kong, S.C., Chiu, M.M., Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers & Education*, 127, 178–189.

Kramer, J. (2007). Is abstraction the key to computing? *Communications of the ACM*, 50(4), 36–42.

Lavy, I. (2021). Learning programming fundamentals via music. *The International Journal of Information and Communication Technology Education (IJICTE)*, 17(2), 68–86

Loewenstein, G. (1994). The psychology of curiosity: A review and reinterpretation. *Psychological Bulletin*, 116(1), 75.

Liao C., (2016). From interdisciplinary to transdisciplinary: An arts-integrated approach to STEAM education. *Art Education*, 69(6), 44–49.
Linnenbrink, E.A., Pintrich, P.R., (2010). The role of self-efficacy beliefs in student engagement and learning in the classroom. *Reading & Writing Quarterly*, 19(2), 119–137.
Littlejohn, A., Margaryan, A., Vojt, G. (2010). Exploring Students' Use of ICT and Expectations of Learning Methods. *Electronic Journal of e-learning*, 8(1), 13–20.
London, R., Ingram, D. (2018). Social isolation in middle school. *School Community Journal*, 28(1), 107–127.
Ludovico, L.A., Mangione, G.R. (2015). Music coding in primary school. In *Smart Education and Smart e-Learning* (pp. 449–458). Springer, Cham.
Margolis, J, Goode, J. (2016). Ten lessons for computer science for all. *ACM Inroads,* 7(4), 52–56.
Manaris, B., Kohn, T. (2016, February). Making music with computers: creative programming in Python. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 711–711).
Marton, F. (1986). Phenomenography – A research approach investigating different understandings of reality. *Journal of Thought*, 21, 28–49.
Master, A., Cheryan, S., Meltzoff, A.N. (2016). Computing whether she belongs: Stereotypes undermine girls' interest and sense of belonging in computer science. *Journal of Educational Psychology*, 108(3), 424.
Metallidou, P., Vlachou, A. (2007). Motivational beliefs, cognitive engagement, and achievement in language and mathematics in elementary school children. *International Journal of Psychology*, 42(1), 2–15.
Olson, I. (2000). *The Arts and Critical Thinking in American Education*. Westport, CT: Bergin & Garvey
Ornek, F. (2008). An overview of a theoretical framework of phenomenography in qualitative education research: An example from physics education research. *Asia-Pacific Forum on Science Learning and Teaching*, 9(2), 1–13.
Paas, F.G.W.C., Van Merriënboer, J.G., Adam, J. (1994). Measurement of cognitive load in instructional research. *Perceptual and Motor Skills*, 79(1), 419–430.
Paas, F., Tuovinen, J.E., Tabbers, H., Van Gerven, P.W. (2003). Cognitive load measurement as a means to advance cognitive load theory. *Educational Psychologist*, 38(1), 63–71.
Papadakis, S., Kalogiannakis, M., Zaranis, N. (2016). Developing fundamental programming concepts and computational thinking with ScratchJr in preschool education: a case study. *International Journal of Mobile Learning and Organization*, 10(3), 187–202.
Petrie, C. (2022). Programming music with Sonic Pi promotes positive attitudes for beginners. *Computers & Education*, 179, 104409.
Pianta, R.C., Belsky, J., Vandergrift, N., Houts, R., Morrison, F.J. (2008). Classroom effects on children's achievement trajectories in elementary school. *American Educational Research Journal*, 45(2), 365–397.
Powers, K. (2011). Going mental: how music education can help develop critical thinking. *Teaching Music*, 18(6), 40–45.
Qian, Y., Lehman, J.D. (2016). Correlates of Success in Introductory Programming: A Study with Middle School Students. *Journal of Education and Learning*, 5(2), 73–83.
Quigley, C.F., Herro, D., Jamil, F.M. (2017). Developing a conceptual model of STEAM teaching practices. *School Science & Mathematics*, 117(1–2), 1–12.
Rantala, T., Määttä, K. (2012). Ten theses of the joy of learning at primary schools. *Early Child Development and Care*, 182(1), 87–105.
Sheridan, S., Williams, P. (2011). Developing Individual Goals, Shared Goals, and the Goals of Others: Dimensions of Constructive Competition in Learning Contexts. *Scandinavian Journal of Educational Research*, 55(2), 145–164
Shneiderman, B. (1977). Teaching programming: A spiral approach to syntax and semantics. *Computers & Education*, 1(4), 193–197.
Siegler, R.S., Wagner A.M. (2005). *Children's Thinking*. Pearson Education/Prentice Hall.
Sjöström, B., Dahlgren, L.O. (2002). Applying phenomenography in nursing research. *Journal of Advanced Nursing*, 40(3), 339–345.
Stringer, E. T. (2013). *Action Research*. Sage publications.
Sweller, J. (1988). Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science*. 12(2), 257–285.
Sweller, J.; van Merrienboer, J, J.G.; Paas, F.G.W.C. (1998). Cognitive Architecture and Instructional Design. *Educational Psychology Review*, 10(3), 251–296.
Takaya, K. (2008). Jerome Bruner's theory of education: From early Bruner to later Bruner. *Interchange*, 39(1), 1–19.

Thomas, T., Davis, T., Kazlauskas, A. (2007). Embedding critical thinking in IS curricula. *Journal of Information Technology Education: Research*, 6, 327–346.

Topoğlu, O. (2014). Critical thinking and music education. *Procedia-Social and Behavioral Sciences*, 116, 2252–2256.

Tsalapatas, H., Heidmann, O., Alimisi, R., Tsalapatas, S., Florou, C., Houstis, E. (2011). Visual programming towards the development of early analytical and critical thinking. In *International Conference on Future of Education*, Konferansında Sunulan Bildiri. Florence, İtalya.

VanderArk, S. D., Ely, D. (1991). Teaching Music functionally: A sociobiologic approach. *Triad*, 56(2), 23–25.

Van Laar, C., Sidanius, J. (2001). Social status and the academic achievement gap: A social dominance perspective. *Social Psychology of Education*, 4(3), 235–258.

Varila, J., Viholainen, T. (2000). *Työnilo Tutkimuksen Kohteeksi* [The joy of working to the target of research] (Research reports 79). Joensuu: University of Joensuu.

White, G.L., Sivitanides, M.P. (2002). A theory of the relationships between cognitive requirements of computer programming languages and programmers' cognitive characteristics. *Journal of Information Systems Education*, 13(1), 59–68.

Yang, T.C., Hwang, G.J., Yang, S.J. and Hwang, G.H. (2015). A two-tier test-based approach to Improving students' computer-programming skills in a web-based learning environment. *Education Technology & Society*, 18(1), 198–210.

Yu, S.N.Y. (2005). *The Effects of Games on the Acquisition of Some Grammatical Features of L2 German on Students' Motivation and on Classroom Atmosphere* (Doctoral dissertation, ACU Research Bank.

Zellner, R.M. (2011). *A Study of the Relationship Between Instrumental Music Education and Critical Thinking in 8th-and 11th-grade Students*. Universal-Publishers.

Zhu, R., Meyers-Levy, J. (2005). Distinguishing between the meanings of music: when background music affects product perceptions. *Journal of Marketing Research*, 42(3), 333–345.

Zimmerman, B.J. (2000). Self-efficacy: An Essential motive to learn. *Contemporary Educational Psychology*, 25, 82–91.

Zentner, M., Grandjean, D., Scherer, K.R. (2008). Emotions evoked by the sound of music: characterization, classification, and measurement. *Emotion*, 8(4), 494.

I. Lavy is an associate professor with tenure at the Academic College of Yezreel Valley and the department head of the Information Systems department. Her Ph.D. dissertation (in the Technion – Israel Institute of Technology) focused on understanding basic concepts in elementary number theory. After finishing a doctorate, she was a post-doctoral research fellow at the Education faculty of Haifa University. Her research interests are in pre-service and mathematics teachers' professional development and the acquisition and understanding of mathematical and computer science concepts. She has published about a hundred and fifty papers and research reports (part of them is in Hebrew).