TEACHING PROGRAMMING IN SECONDARY EDUCATION THROUGH SOUND

Theofani S. Sklirou
Department of Informatics and Telecommunications
University of Peloponnese
Tripoli, Greece
fanisklirou@yahoo.gr

Areti Andreopoulou
Laboratory of Music Acoustics and Technology (LabMAT)
National and Kapodistrian University of Athens
a.andreopoulou@music.uoa.gr

Anastasia Georgaki
Laboratory of Music Acoustics and Technology (LabMAT)
National and Kapodistrian University of Athens
georgaki@music.uoa.gr

ABSTRACT

It is considered hard to teach programming in secondary education, while following the steps of the provided curriculum. However, when teaching is supported by suitable methodologies, learning can be ameliorated. Under this premise, this paper discusses a different teaching approach to programming in secondary education and examines the potential benefit of sound-alerts as a complementary teaching tool. Such alerts were created by pairing sound stimuli to specific programming actions and operations. Both the selection of sound stimuli as well as the potential impact of the use of sound alerts on programming were evaluated through perceptual studies. Results showed that participants preferred synthesized to natural (pre-recorded) stimuli for all types of alerts. It was also revealed that users prefer sound-alerts associated to pending actions, errors, and successful code execution, over alerts highlighting a step-by-step execution of the code.

1. INTRODUCTION

According to a popular definition, programming is the process of writing, testing, debugging/troubleshooting, and maintaining the source-code of computer programs [1]. In Greek secondary education, programming courses were introduced to the curriculum 25 years ago. Since then, students have been confronted with problems concerning human computer interaction through coding, as the latter requires a precise way of thinking realized through specific syntax [2, 3, 4]. Nevertheless, while the most common difficulties that students encounter when learning how to program have been identified, clear strategies for addressing them still remain to be established.

High-school students should be taught programming concepts independently of specific applications and programming languages [5, 6]. They all have different needs and difficulties, which can be divided into 5 categories [7]: 1) orientation: discovering the usefulness and benefits of programming, 2) notional machine (the general properties of the machine): realizing how the behavior of the physical machine relates to the notional machine, 3) notation: facing problems related to syntax and semantics, 4) structures: understanding the schemas or plans that can be used to reach small-scale goals (e.g., using a loop), and 5) mastering the pragmatics of programming: learning the skill to specify, develop, test, and debug a program using the available tools.

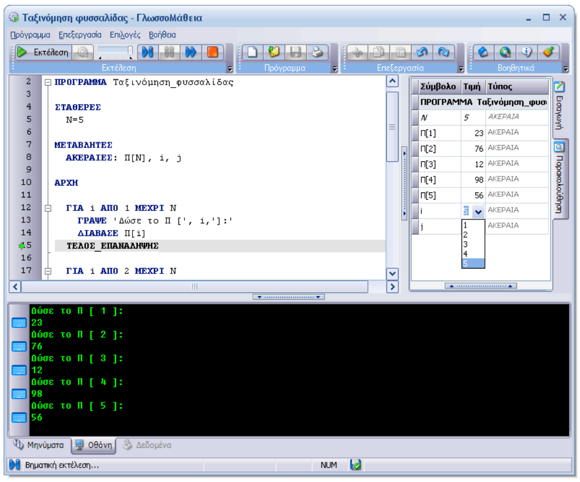
Pea has identified certain persistent conceptual language-independent "bugs" in how novices program and understand coding [8]. Students believe that computers "go beyond the information given" in a program. In addition, it has been observed that several of them fail to "translate" a conceptual solution to a problem into the correct code [9]. The reason might be that students are not trained to transform conceptual intuitions into code. Such obstacles, could be overcome by helping students develop problem-solving skills in addition to logical reasoning.

It is known that Artificial languages have a limited vocabulary compared to natural ones. Yet, teachers use natural languages to decode and communicate the meaning of programming operations. It has been shown that multimodal interactions facilitate the understanding of programming concepts [10, 11]. The most common practices in Greek schools involve environments with audio-visual feedback [12, 13]. This paper explores the use of sound-alerts as a complementary tool to programming courses, and discusses their potential impact on the students' problem solving skills development.

2. PROGRAMMING IN SECONDARY EDUCATION: AN OVERVIEW OF EDUCATIONAL TOOLS

This section presents an overview of the programming methods used in secondary education, and thoughts around programming environments, in general. The first attempts to present programming in a more engaging manner started in the early '70s. Among the most popular environments were *Logo* and its derivates *Kodu* and *Alice* [14]. Some of these proposals promoted the use of visual or virtual programming languages and the simulation of a dynamic auralization of the program execution [15].

Γλωσσομάθεια (Glossomatheia) is a pseudocode based programming environment, written in *Pascal*, used in secondary education (high-school) in Greece. [16, 17, 18]. It is a training package, developed with a focus on laboratory support courses related to the cultivation of algorithmic and analytical thinking, and the development



```
ΠΡΟΓΡΑΜΜΑ Ταξινόμηση_φυσσαλίδας

ΣΤΑΘΕΡΕΣ
  N=5

ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Π[N], i, j

ΑΡΧΗ

  ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ N
    ΓΡΑΨΕ 'Δώσε το Π [', i,']:'
    ΔΙΑΒΑΣΕ Π[i]
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

  ΓΙΑ i ΑΠΟ 2 ΜΕΧΡΙ N
```

Σύμβολο	Τιμή	Τύπος
ΠΡΟΓΡΑΜΜΑ Ταξινόμηση_φυσ		
N	5	ΑΚΕΡΑΙΑ
Π[1]	23	ΑΚΕΡΑΙΑ
Π[2]	76	ΑΚΕΡΑΙΑ
Π[3]	12	ΑΚΕΡΑΙΑ
Π[4]	98	ΑΚΕΡΑΙΑ
Π[5]	56	ΑΚΕΡΑΙΑ
i	5	ΑΚΕΡΑΙΑ
j		ΑΚΕΡΑΙΑ

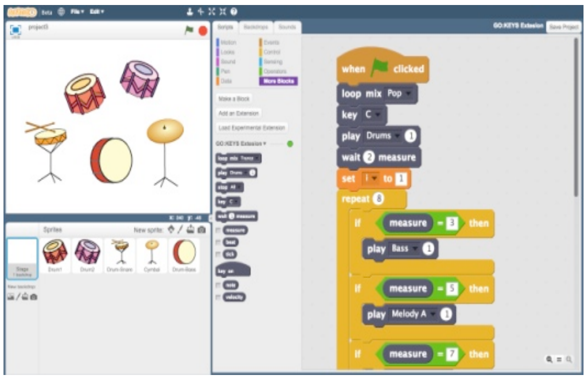
```
Δώσε το Π [ 1 ]:
23
Δώσε το Π [ 2 ]:
76
Δώσε το Π [ 3 ]:
12
Δώσε το Π [ 4 ]:
98
Δώσε το Π [ 5 ]:
56
```

Figure 1. The *Glossomatheia* programming environment

of methodological skills for students. An example of the software can be found in Figure 1. As can be seen, it relies heavily on visual cues, using written messages as the primary communication method with the user. The work presented in this paper is using *Glossomatheia* as a basis for the evaluation of the effect of auditory cues on computer programming comprehension.

The closest approach to an educational tool for programming employing audio cues is *Scratch*. *Scratch* is used in education and entertainment, and is suitable for students at a starting age of 8 years old [19]. Students can easily create interactive stories, animations, computer games, music and digital art. Sensors can also be used with Pico board, a piece of hardware, allowing the interaction of Scratch projects with the outside world. The system offers support for music blocks controlling loop-sets, play-back etc. [20].

Peep is a Network Auralizer that replaces visual monitoring with a sonic `ecology' of natural sounds. Each sound-type represents a specific kind of network event [21]. The idea of testing auditory feedback using natural sounds in the system presented in this paper was highly motivated by Peep. After visualization, information is transmitted through sound using perceptually relevant parameters, such as intensity and frequency [22].



```
when clicked
loop mix Pop
key C
play Drums 1
wait 2 measure
set i to 1
repeat 8
  if measure = 3 then
    play Bass 1
  if measure = 5 then
    play Melody A 1
  if measure = 7 then
```

Figure 2. The Scrach X environment

3. TEACHING PROGRAMMING USING BIMODAL INTERACTIONS

According to [23], students in Greek high-schools learn only basic elements of programming, due to the inefficient and, in some cases, outdated teaching methods, and the absence of an interconnection between education and the industry. In an attempt to alleviate the former, this paper discusses a methodology for teaching programming through sound-alerts stimulating psychoacoustic perception. The explored sound quality characteristics include, but are not limited to, pitch, tempo, rhythm, timber, loudness, roughness, and sharpness etc. [24]. It has been shown that the use of physiological measures sensitive to attention and arousal, in conjunction with behavioral and subjective measures can lead to the design of auditory warnings that produce a sense of diversity of programming commands [25]. Given that acoustic and visual memory make up 90% of the sensory memory, it can be hypothesized that students receiving both visual and acoustic feedback will gain a deeper understanding of programming structures.

The long-term goal of this project is to assist students understand different algorithmic procedures, such as relational and arithmetic operations, through bimodal (visual and aural) feedback. The underlying hypothesis is that through visual and aural interactions students will comprehend programming structures more effectively.

3.1 Stimulus selection

One crucial component in the design of a bimodal programming environment is the selection of the utilized sound stimuli, which, in order to be conducive to the students' learning, should reflect in a clear and concise manner the algorithmic action they correspond to. In addition, they should also reflect the aesthetics of the target group, in this case consisting of senior high-school students.

Hence, a preliminary study was conducted aiming at the selection of the most appropriate sound stimuli, given the aforementioned criteria. Two different categories of sounds were tested, natural (environmental / ecological) and synthesized. Ecological sounds have richer timber and pitch variation than synthesized ones and their use is important.

Five procedures and/or operations were selected for evaluation: a) reading data from keyboard b) successful data assignment, c) error, d) arithmetic operation, and e) relational operation. For each procedure, a pair of sound stimuli were selected (1 natural and 1 synthesized), such that they shared common auditory characteristics in terms of pitch, timbre, speed, and contour. Participants were asked to select the best fitting sound alerts for each of tested procedures.

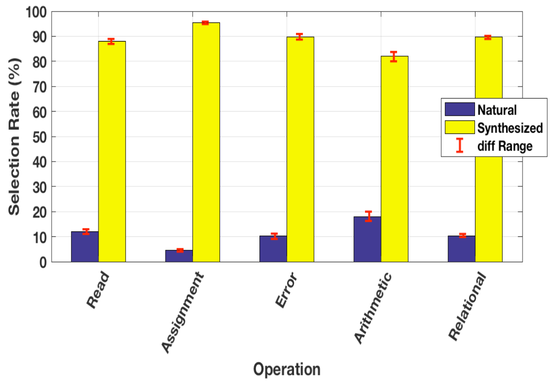


Figure 3. Preliminary test result overview. Bars correspond to the selection rate of each sound stimulus grouped per process and averaged across test repetitions. Variations between repetitions are marked with error-bars.

3.2 Protocol overview

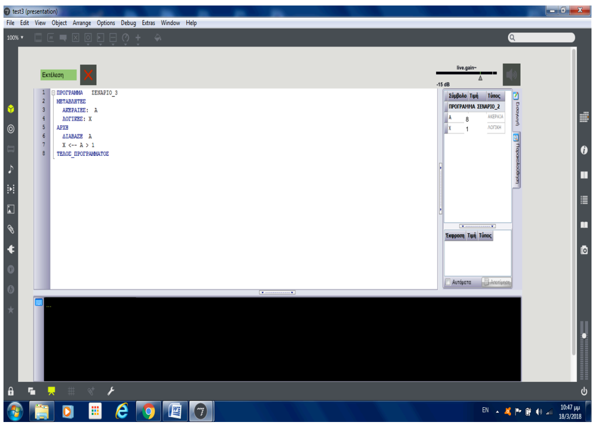
146 senior high-school students (67 male), 16 to 17 years old, participated in this preliminary study. Participants were divided into 8 groups. Groups were presented with a pair of stimuli (sound-alerts) for each of the 5 coding procedures, and were asked to select the alert that best fitted each procedure in a 2AFC task with no repetitions. Each group took the test twice, once in the beginning and a second time at the end of a class, to evaluate response repeatability. The approximate duration of each test was 5 minutes. Between groups both the order of stimuli and test procedures were fully randomized, but remained the same within groups for practical purposes.

3.3 Preliminary Study Results

The stimulus type selection results of the preliminary study are summarized in Figure 3. As can be seen, the vast majority of the students (82% - 96%) preferred synthesized alert sounds over natural ones. No significant deviations were observed as a function of stimulus and/or procedure presentation order. Student preference, remained roughly unchanged across the 2 test repetitions (see Figure 3 error-bars). Variations were smaller than 3% across all tested procedures. These results highlight that synthesized sound-alerts are highly preferred for such types of interactions.

4. EVALUATION STUDY

Following the preliminary study, a second experiment was conducted to assess the effectiveness of sound-alerts as a complementary tool for teaching computer programming. 53 senior high-school students (24 male), who have previously participated in the preliminary study, volunteered to participate. All of them were taking, at the time, programming classes at school using *Glossomatheia*.



```
ΠΡΟΓΡΑΜΜΑ   ΣΕΝΑΡΙΟ_3
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ:  Α
  ΛΟΓΙΚΕΣ: Χ
ΑΡΧΗ
  ΔΙΑΒΑΣΕ  Α
  Χ <-- Α > 1
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Σύμβολο	Τιμή	Τύπος
ΠΡΟΓΡΑΜΜΑ ΣΕΝΑΡΙΟ_2		
Α	8	ΑΚΕΡΑΙΑ
Χ	1	ΛΟΓΙΚΗ

Figure 4. Max/MSP test interface (evaluation study)

4.1 Experimental Protocol

Participants were presented with 3 ready-to-run programming scenarios in *Glossomatheia* and *Max/MSP*. The *Max/MSP* patch was designed to have an identical user-interface to *Glossomatheia*. Its only difference was that it complemented visual feedback with sound-alerts. The sound stimuli utilized for the alerts were selected from the most preferred sounds of the preliminary study.

Participants were allowed to interact with both environments and were afterwards asked to fill a questionnaire evaluating the effect of auditory feedback on the comprehension of the code functionality. For each tested scenario, user ratings were collected on a 5AFC Linkert scale with the following anchors: no affect, minor affect, neutral, moderate affect, major effect. The questionnaire concluded with a "general comments" section, where participants could share their thoughts and feedback on the tested system.

4.2 Teaching scenarios

The following 3 teaching scenarios were tested:

4.2.1. Scenario 1: Data entry

The code performed an assignment of a numerical value to a pre-defined variable. Upon execution, the program waited for user-input from the keyboard. If the input value was numeric an assignment was performed and the program concluded. Yet, if user-input was not numeric, the code returned an error and waited for new input.

When this scenario was evaluated in *Glossomatheia*, the waiting time for user-input was indicated by a flashing cursor on the computer screen. In the *Max/MSP* environment, except for the flashing cursor, users heard a sound-alert informing them of a pending action. If the user-input was numeric, *Glossomatheia,* printed the assignment on screen and the code concluded, while Max/MSP complemented the visualization with a sound-alert indicating successful assignment. If user-input was not numeric, *Glossomatheia* printed an error message on screen, while *Max/MSP* produced an additional sound-alert, indicating an erroneous action.

Operation	No affect	Minor affect	Neutral	Moderate affect	Major affect
Read	11,32%	0,00%	24,53%	30,19%	33,96%
Assignment	11,32%	1,89%	7,55%	35,85%	43,40%
Error	9,43%	3,77%	15,09%	39,62%	32,08%
Arithmetic	13,21%	15,09%	37,74%	16,98%	16,98%
Relational	9,43%	18,87%	35,85%	28,30%	7,55%

Table 1. Participant evaluations of the effect of sound-alerts on code comprehension

4.2.2. Scenario 2: Arithmetic operation

The code performed an assignment of a numerical value to a pre-defined variable followed by a simple numerical operation (addition to a constant). The first part concerning the assignment of user-input to a variable was identical to scenario 1. Hence the code worked exactly as described in Section 4.2.1, and both *Glossomatheia* and *Max/MSP* alerts remained the same. When the arithmetic operation was executed *Glossomatheia* printed the result on the computer screen, while the *Max/MSP* test-environment produced a complementary sound-alert indicating that an arithmetic operation had been performed.

4.2.3. Scenario 3: Relational operation

The code performed an assignment of a numerical value to a pre-defined variable followed by a simple relational operation (comparison of the input to the numerical value of 1). The first part concerning the assignment of user input to a variable was identical to scenario 1. Hence the code worked exactly as described in Section 4.2.1, and both *Glossomatheia* and *Max/MSP* code alerts remained the same. When the relational operation was executed *Glossomatheia* printed the boolean result on the computer screen, while the *Max/MSP* test environment produced a complementary sound-alert indicating that a relational operation had been performed.

4.3 Results

Participant evaluations of the effect of sound-alerts on code comprehension are summarized in Table 1. As can be seen, more than 60% of the assessors indicated that the use of sound-alerts had a positive effect (moderate or major) on code comprehension of the following operations/actions: waiting for data input (read), successful assignment of data to a variable, and erroneous code execution (error). In addition, it appears that the use of sound-alerts had no effect (neutral) to users in the case of arithmetic and relational operations, fact which could be interpreted in two different ways: either users preferred sound-alerts for events pertinent to the correct or erroneous execution of the code and for notifications of pending actions, or the specific experiment design and rating questions were not appropriate for testing the effectiveness of sound-alerts on other types of operations.
It should also be noted that out of the 53 participants less than 12% indicated that the complementary use of sound-alerts had no effect on code comprehension (Table 1). The remaining students did feel that the auditory cues had an impact on code understanding. This observation is also reflected on Figure 5, which plots user evaluations averaged across the three tested scenarios. As can be seen, approximately 57% of the students felt that the effect was moderate or major compared to 19% who felt that the effect was minor or non-existent.

5. CONCLUSIONS & FUTURE WORK

This paper discussed the potential benefits of using auditory cues (sound-alerts) as a complementary tool for teaching programming in secondary education. The work was based on the hypothesis that bimodal user interactions could positively impact the students' development of problem solving skills, and improve their comprehensions of programming code. Two studies were presented. The first assessed the type of sounds which would be preferable for such a task, while the second whether or not the use of sound-alerts affects code comprehension.

Two different sound categories were considered: recorded excerpts of bird sounds (natural sounds) and electronic sounds from synthesizers (synthesized sounds). Five computational procedures and/or operations were evaluated (reading data from keyboard, successful data assignment, error, arithmetic operation, and relational operation. For each procedure, a pair of sound stimuli were selected and paired to a sound from each of the two categories. The sound pairs shared common musical and psychoacoustic properties, such as the perceived pitch and loudness [24, 27], while varying in terms of timbre.

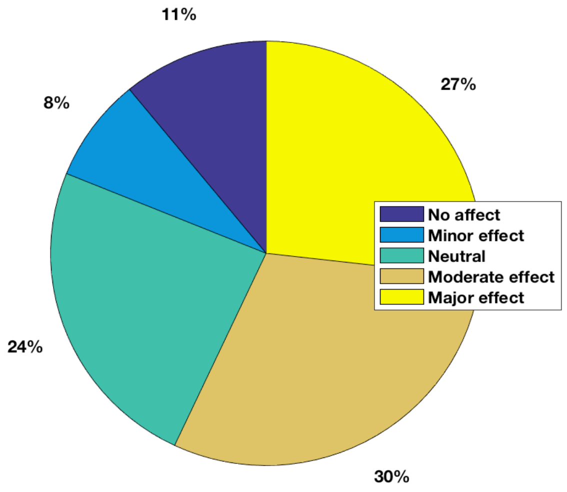


Figure 5. Overall evaluation of the use of sound-alerts on code comprehension averaged across all gteaching scenarios.

Auditory display connects psychoacoustics with cognition based on sound attributes. The most important aspects in auditory design relate to whether the listener can hear changes of particular parameters in a given sound. Timbre is a catch-all term in both psychoacoustics and auditory display, often used to imply various sound attributes. The ability to distinguish sounds of different timbres has been important in mapping data to audio. On the other hand, pitch is the most commonly used auditory display dimension. This is because it is easy to manipulate and, generally speaking, changes in pitch are easily perceived. [26].

Participants showed very strong and consistent preference towards synthesized sounds, rejecting natural ones almost unanimously across all tested procedures. This can be attributed to the fact that the context of the selected natural sounds (bird voices) could not be directly associated with the technical concept of the task.

In the second study, participants had to evaluate the effect of sound-alerts on code comprehension, given three programming scenarios. Overall, students rated the use of auditory feedback positively. 57% of them indicated that the cues had a moderate to major effect on their understanding, while only 11% indicated no effect at all. Interestingly enough, participants showed stronger preference for sound-alerts related to pending activities and correct or erroneous executions of the code than to other operations. This can be related to the fact that sound alerts work well as memory boost [28], hence notifying users of any code-related events that require action. Our interpretation of the ratings is further supported by some of the provided written feedback. For example, some students wrote: "I prefer the sounds for success and error", "the pending action sound helped me understand that was time to input some data", "sound alerts for numerical and relational operations were not so important".

Certain participants indicated that a combination of sound-alerts and voice messages could be effective. This is certainly a route worth exploring as this project advances. In moving forward, the first step would be to include sound-alerts for more procedures and operations, and test them against more complex teaching scenarios. Such will include looping, conditional statements, data sorting, element searching etc.

Acknowledgments

The authors would like to thank all students who participated on the two studies presented in this paper.

6. REFERENCES

[1] M. Saeli, J. Perrenet, W. Jochems, B. Zwaneved, “Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective”, 2011, Retrieved from https://files.eric.ed.gov/fulltext/EJ1064282.pdf, [Last visited 05/04/2018]

[2] N. Avouris, “Introduction to Human Communication – Computer”, Athens: Diavlos, 2000.

[3] M. Grigoriadou, A. Gogolou, E. Gouli, K. Glezos, M. Boubouka, K. Papanikolaou, C. Tsagkanou, E. Kanidis, D. Dukakis, S. Fragkou, and H. Verginis, “Teaching Approaches and Tools for teaching IT”, Athens: New Technologies, 2009.

[4] A. Robins, J. Rountree and N. Rountree, “Learning and Teaching Programming: A Review and Discussion. Computer Science Education”, 2003.

[5] C. Stephenson, J. Gal-Ezer, B. Haberman and A. Verno, “The New Educational Imperative: Improving High School Computer Science Education” (Rep. No. Final Report of the CSTA Curriculum Improvement Task Force – February 2005).

[6] P. Szlávi, and L. Zsakó, “Programming versus application”, In: Mittermeir, R.T. (Ed.), ISSEP 2006, LNCS 4226, 2006, 48–58.

[7] B. Du Boulay, “Some difficulties of learning to program”, In: Soloway, E., Spohrer, J.C. (Eds.), Studying the Novice Programmer, London, Lawrence Erlbaum Associates, 1989, 283–299.

[8] R.D. Pea, “Language-independent conceptual “bugs” in novice programming”, Journal of Educational Computing Research, 1986, 2, 25–36.

[9] M. Weigend, “From intuition to program. Programming versus application”, In: Mittermeir, 2006.

[10] K. Tsolakidis and M. Fokidis, “Virtual reality in education”, Athens, 2007.

[11] A. Vakaloudi, “Teaching and learning with new technologies theory and practice”, Athens: Patakis, 2003.

[12] S. Aslanidou, “Educational Technology, From the audiovisual in the digital treatment”, 2010.

[13] C. Kelleher and R. Pausch, “Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers”, ACM-Computer-Surveys, 2005.

[14] Noss, R., “Children Learning Logo Programming” Interim Report No. 2 of the Chiltern Logo Project, Advisory Unit for Computer Based Education, Hatfield, United Kingdom, 1984.

[15] Beanz, the magazine for kids, code, and computer science (https://www.kidscodecs.com/resources/programming/education/), [Last visited 05/04/2018]

[16] Spinet (http://spinet.gr/glossomatheia/), [Last visited 05/04/2018]

[17] Algorithmos (http://www.algorithmos.gr/glossomatheia.html), [Last visited 05/04/2018]

[18] Ebooks.edu.gr (http://ebooks.edu.gr/courses/DSGL-C101/document/4c65902ff3dk/4e52d483egdp/4e52e406mj6l.pdf), [Last visited 05/04/2018]

[19] Scratch Wiki (https://wiki.scratch.mit.edu/wiki/ScratchX), [Last visited 05/04/2018]

[20] P. Kirn, "Roland and MIT want to use music to teach kids programming", 2018, Retrieved from http://cdm.link/2018/01/roland-mit-want-use-music-teach-kids-programming/, [Last visited 05/04/2018]

[21] M. Gilfix, A. Couch, "Peep (The Network Auralizer): Monitoring Your Network With Sound", https://www.usenix.org/legacy/events/lisa00/gilfix/gilfix_html/, [Last visited 05/04/2018]

[22] V. Cerf, "Communications of the ACM. The sound of programming". Retrieved from https://cacm.acm.org/magazines/2018/4/226379-the-sound-of-programming/fulltext, [Last visited 05/04/2018]

[23] I. Milne, G. Rowe, "Difficulties in Learning and Teaching Programming—Views of Students and Tutors", 2002, Retrieved from http://www.swisseduc.ch/informatik-didaktik/programmieren-lernen/docs/milne.pdf, [Last visited 05/04/2018]

[24] K. Genuit, "Sound quality in environment: "Psychoacoustic mapping", 2004, https://doi.org/10.1121/1.4785535

[25] J. L. Burt, D. S. Bartolome, D. W. Burdette and J. R. Comstock JR, "A psychophysiological evaluation of the perceived urgency of auditory warning signals", 2007, Pages 2327-2340, https://doi.org/10.1080/00140139508925271

[26] B. Walker and G. Kramer, "Ecological Psychoacoustics and Auditory Displays: Hearing, Grouping, and Meaning Making", Retrieved from http://sonify.psych.gatech.edu/~walkerb/publications/pdfs/2004WalkerKramer-Ecological_psychoacoustics.pdf [Last visited 31/05/2018]

[27] H. Fastl, "Psychoacoustic basis of sound quality evaluation and sound engineering", 2006, Retrieved from https://mediatum.ub.tum.de/doc/1138486/file.pdf, [Last visited 05/04/2018]

[28] D. Yarbrough, " Sound the alarm: how sounds affect our memory and emotions", 2017, Retrieved from https://www.voxmagazine.com/music/sound-the-alarm-how-sounds-affect-our-memory-and-emotions/article_153c4146-be25-11e7-b9ab-8b1620bcc28d.html