muco: A MUSIC COMPUTING LEARNING APPLICATION

Patricia Hu[1, 4], Oliver Hödl[1, 2], Peter Reichl[1], Fares Kayali[2], Iris Eibensteiner[3], Bernhard Taufner[3], Sigrid Schefer-Wenzl[3], and Igor Miladinovic[3]

[1]Cooperative Systems Research Group, University of Vienna, Austria firstname.lastname@univie.ac.at
[2]Centre for Teacher Education, University of Vienna, Austria firstname.lastname@univie.ac.at
[3]Software Design and Engineering Department, University of Applied Sciences Campus Vienna, Austria firstname.lastname@fh-campuswien.ac.at
[4]Institute of Computational Perception, Johannes Kepler University Linz, Austria patricia.hu@jku.at

ABSTRACT

Following a computational approach to music creation can serve as a transdisciplinary bridge between computer science, music and education by providing rich opportunities for teaching computing skills in interdisciplinary contexts. Making use of such learning contexts is one of the cornerstones of STEAM (STEM + Arts) education, a pedagogic approach focusing on integrating the arts and humanities into STEM fields. Despite numerous benefits reported in the literature, the number of STEAM programs integrated into formal learning contexts remains limited due to various integration challenges. In addressing these challenges, we developed and evaluated a gamified music computing learning application called muco, using the MVC software architecture along with React as the front-end, and Node and MariaDB as back-end technologies. The goal of the application is to make programming concepts more accessible and comprehensible to a broader public, and to simultaneously spark interest in music and music making. Making use of game mechanics, muco is conceptualised to teach computational thinking, and more specifically, introductory programming, in non-formal learning contexts. The usability of the application and impact of interdisciplinary learning is evaluated by means of both formative and summative testing. Results show that the application provides an engaging learning environment which encourages the exploration of both the computing and music domain.

1. INTRODUCTION

There are many parallels between musical and software artefacts. To a certain extent, music can be considered 'organised sound', containing hierarchical structures formed by musical events of different levels, each varying in their degree of abstraction [1, 2]. These inherent structural relationships and the hierarchical nature in and between musical patterns offer ample grounds to approach music in an algorithmic or computational manner [3, 4].

The idea of adopting an algorithmic approach to music and music making has in fact been explored across millennia and across cultures [5]. More recently, a number of researchers have also pointed to the potential of adopting an analytical approach to music to enrich existing didactic methods in either music or computer science disciplines [4, 6]. In the latter case in particular, a systematic and structured engagement with music can provide a rich pedagogic context for exploring the creative sides of computing [7, 8].

The field of computational thinking within educational research and practice is steadily growing, and an increasing number of countries worldwide are integrating programming education into their national school curricula [9, 10]. As suggested above, music and music education (involving various aspects of music such as understanding, analysis, composition etc.) provide many opportunities for teaching computational thinking and programming concepts in a creative context. This is especially true in the area of STEAM (STEM + Arts) education.

The muco web application builds on the premise of bridging two domains - computing and music - to explore and exploit interdisciplinary potentials for knowledge transfer and learning. The application is intended to be suitable for beginners of both fields, thus requiring no prior knowledge in either music or programming.

Likewise, striving to advance the field of STEAM education, the muco application is designed to be used in non-formal, non-structured pedagogic settings. To encourage learning persistence in such contexts, the application leverages the potential of gamification techniques in order to promote student engagement and motivation, both of which have been identified as key factors in any learning process [11]. To this end, the muco application features selected game elements.

The remainder of this work is structured as follows: after providing a succinct introduction to the relevant subject areas in section 2, section 3 illustrates the applied research approach. Next, section 4 presents the muco application from both a technical and content-related point of view. Section 5 then describes the evaluation of the application, with results being presented and discussed in section 6. Lastly, section 7 concludes with a summary of lessons learned and areas for future research.

On a final note, it should be noted that the results pre-

sented herein do not claim to be exhaustive. Rather, the present work strives to be explorative in nature, pursuing the overall goal of making computing disciplines more accessible and comprehensible to a broader audience, while simultaneously sparking interest in music and making music.

2. RELATED WORK

2.1 STEAM Education

STEAM (as an acronym of the included disciplines Science, Technology, Engineering, Arts and Mathematics) is a transdisciplinary pedagogic approach focusing on the incorporation of liberal arts and the humanities into classical STEM education. As an emerging educational practice, it is increasingly gaining importance and recognition within the pedagogic community [12–14].

The literature on STEAM education reports a wide range of positive outcomes, with multiple empirical studies demonstrating improved content acquisition and conceptual understanding of scientific topics [15–17], increased task- or skill-related proficiency [16,18,19] and overall improved motivation and attitudes towards a specific STEM discipline [8, 14, 16, 19]. Despite these positive effects, however, researchers have identified recurring issues such as a lack of shared understanding of the STEAM concept and differing opinions on the nature and role of arts integration [13].

Another challenge reported in the literature relates to practical obstacles hindering effective STEAM program implementation. In this context, four major problem sources have been identified by Herro et al. [20], which relate to adequate pacing and time allocation of STEAM teaching units, observed student struggles with self-directed learning, challenges related to content and discipline alignment as well as teacher collaboration, and issues resulting from inflexible school policies.

2.2 Computational Thinking and Music

The concept of computational thinking emerged in the 1950s and 1960s in the field of computer science as 'algorithmic thinking'. In the traditional sense, the term can be described as a habit of mind of designing and crafting useful and effective software programs [21].

Since then, a variety of definitions of the term have been proposed [9, 22], most of which involve the mental concepts of problem reformulation, abstraction and decomposition. Referring to the relation of computational thinking to programming, it is clear that both are not the same, but being able to think computationally does facilitate the transition of problem-solving methods to formalized programming languages [23]. Though the definition and taxonomy of computational thinking is still evolving [24], the concept in itself has been met with great enthusiasm by educators as well as academics, not least due to its contribution in attracting attention to the field of computer science [25, 26].

Likewise, pedagogic scholars and practitioners from both computing and music fields have pointed to the educational potential of interdisciplinary overlaps. As indicated above, musical pieces and software programs mirror each other in a number of aspects. Building on the idea of approaching music in an analytic manner, a number of researchers have emphasized parallels not only between musical and software artefacts, but also between the thought processes involved in learning and making music and software [3, 4, 6–8].

2.3 Digital Education and Gamification

Digital learning refers to any learning framework or system involving the use of information and communication technologies (ICT), including electronic learning (e-Learning) and mobile learning (m-Learning) [27]. Gamification implies the incorporation of game thinking and game design techniques to conventionally non-game activities. In the educational context, this is often applied to increase student engagement and improve learning outcomes overall [28]. Making use of its resources and innovative technologies, digital learning offers an ideal environment for the integration of game elements and mechanics.

The idea of using gamification to enhance student engagement and motivation has been applied and studied in formal and informal education for some time [29]. Both digital learning and the application of gamification therein have also led to increased focus and emphasis on the concept of self-directedness or self-directed learning, which stresses the ability of the student to plan and manage resources independently and apply critical problem-solving [30].

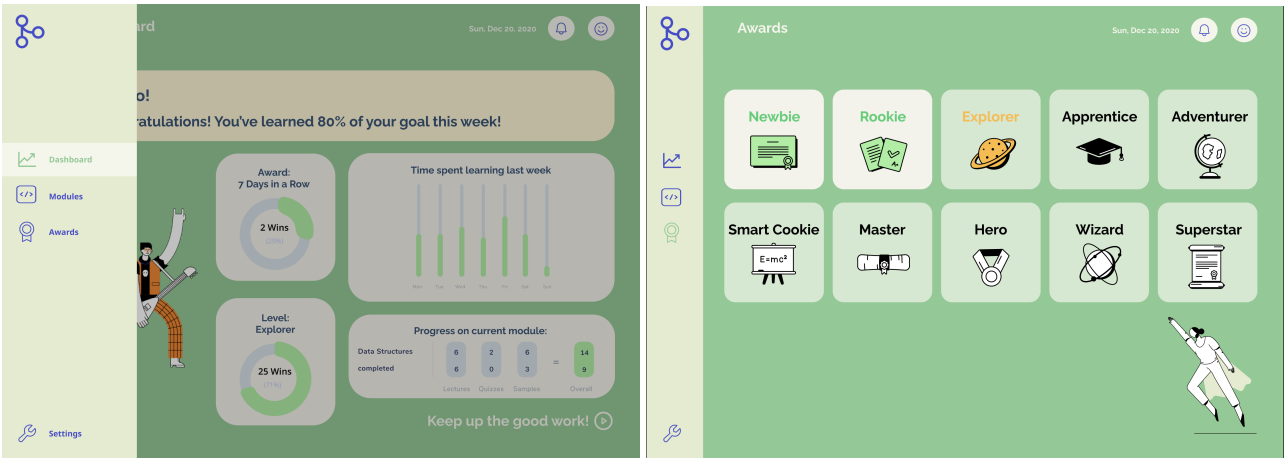
3. METHODOLOGY

Given the interdisciplinary nature of the present work, a mixed-methods approach was applied for the different steps involved in the design, implementation and evaluation of the `muco` learning application.

First, an iterative prototyping approach was followed for the design and implementation of the `muco` application, with each iteration realising a higher functional scope than the previous one. Next, the application was evaluated both qualitatively and quantitatively by means of formative and summative usability testing.

Formative testing was performed with an earlier prototype version of the `muco` application, specifically one which did not involve any gamification mechanics. These early-stage studies aimed at evaluating the general concept and learning content of the `muco` application and were conducted by means of semi-structured expert interviews. Interviews provide an appropriate context to ask open-ended questions on a broad range of topics which makes them suitable for gaining a deep and nuanced understanding of a problem [31].

Summative testing tends to focus on task measurements and related quantitative metrics, and is therefore more appropriate for more advanced development stages [32]. In the context of the present work, two summative usability studies were conducted to evaluate the final prototype version, which included game mechanics.



(a) (b)

Figure 1: (a) Dashboard (with open sidebar) and (b) Collectible Rewards in the muco application

4. MUCO APPLICATION

4.1 Design Rationale

A a starting point, a low-fidelity wireframe was created, which defined the fundamental structure and navigation of the application. Figure 1 presents excerpts from this wireframe. The design process was guided by Nielsen's *Ten Usability Heuristics* [33] and Shneiderman's *Eight Golden Rule for Interface Design* [34], both of which can be considered best practice principles in the realm of interface and usability design.

The most relevant functional requirements included the choice of a synthesis engine which supports the use of a general purpose language for teaching programming concepts and a browser-supported text-based editor serving as the development environment. Taking into account the design rational of conceptualising muco for use in non-formal, self-directed learning contexts, a rewards collection and dynamic module unlocking feature along with selected game elements were also included in the functional scope.

4.2 Technology Stack

The muco application is realised as a web application using React [1] in the Front-End, Node [2] and MariaDB [3] in the Back-End, and the Model-View-Controller (MVC) design pattern for the software architecture. The implementation of the muco application was preceded by the definition of key design and functional requirements, both of which formed the basis for the choice of application type, technology stack and software architecture.

With regards to the underlying synthesis engine, the Tone.js [4] library was chosen as it supports coding music in JavaScript, a high-level general purpose language that is both industry-relevant and beginner-friendly [35]. Tone.js is a web-based audio framework enabling interactive music making in the browser. The library provides basic objects and methods for creating low-level oscillators and synthesizers. In addition, the library also features higher-level monophonic and polyphonic instruments and samples, as well as methods for adding effects and controlling the scheduling of multiple audio events. For the realisation of a text-based development environment, both the Ace Editor [5] and CodePen pens [6] were used, with the former serving as read-only editors for presentation and illustration purposes and the latter providing an interactive development environment in which the user could try out (and run) new programming concepts.

4.3 Learning Modules

From a content and subject matter point of view, the main focus was centered on the learning modules, which were developed in two stages. In a first step, the structure for all modules was drafted as follows:

- *Theory* component: The first part of each learning module presents sound and music related concepts. They are intended to serve as theoretical constructs that can be used to illustrate programming concepts that are presented in the subsequent component.
- *Code* component: In the second part of each learning module, specific programming concepts are introduced. Each concept is explained and illustrated by means of a concrete code example that is displayed in an editor component. Each of the code examples also produces an audio output, which the user can trigger by running the code.
- *Try It Yourself* component: In the third part of each learning module, the user is encouraged to try out the previously introduced concept by him/herself and/or test him/herself on the content of the current module. To this end, this component provides either an interactive development environment, or a gamified test element.

[1] https://reactjs.org/ (accessed 31-10-21)
[2] https://nodejs.org (accessed 31-10-21)
[3] https://mariadb.org/ (accessed 31-10-21)
[4] https://tonejs.github.io/ (accessed 31-10-21)
[5] https://ace.c9.io/ (accessed 31-10-21)
[6] https://codepen.io/ (accessed 31-10-21)

Figure 2: The general structure of a learning module as illustrated by the first module, titled 'Learning By Playing'. The yellow area contains the *Theory*, the orange area the *Code*, and the blue area the *Try It Yourself* component.

Figure 2 presents the general structure described above, as illustrated by the introductory module titled 'Learning by Playing'.

In a second step, the actual structure of the learning contents and teaching objectives with regards to the music and programming concepts were defined and formulated. The learning content of the application was formalised in such a way to be suitable for beginners in either field, thus requiring neither musical nor programming prior knowledge.

Table 1 provides an overview of the learning modules provided in the application, including a list of the music and programming concepts covered in each module.

4.4 Game Mechanics

With the aim of increasing student engagement and motivation in self-directed learning contexts, a rewards collection and dynamic module unlocking feature were integrated into the `muco` application along with selected game elements. The game elements were implemented in the form of micro-assignments in the *Try It Yourself* components of each learning module, either as quizzes consisting of multiple-choice and/or single choice questions or sorting tasks in which the user is asked to bring an unsorted code script into the right order. To encourage the user to continue learning, these micro-assignments were linked to the module unlocking and rewards collection mechanism in such a way that requires the user to successfully pass an assignment in order to unlock a new module. In addition, the rewards were conceptualised to reflect different skill levels through corresponding badge titles.

5. EVALUATION

5.1 Formative Testing

Taking into account the interdisciplinary aspect of the `muco` application, domain experts from the field of music were selected as interview partners. To this end, five formally trained musicians were invited as study participants, among them two performing musicians, one music education student, one music teacher and one experimental media and sound artist. Except for one all participants were in their twenties (M27, M27, F22, M24, M35). Furthermore, except for the experimental musician, none had any prior programming knowledge or experience.

The interviews were carried out in a semi-structured manner and followed a conversational rather than inquisitorial communication style. A time frame of two to three hours was allocated for each interview to ensure sufficient time for both hands-on exploration and follow-up questions. At the beginning of each interview, a short introduction on the research context was given and informed consent was obtained regarding the collection and further analysis of the data acquired in the course of the study. Next, each participant was asked to introduce her/himself. This was followed by questions on expectations towards the `muco` application, which aimed at capturing the interviewees' first impression of the general concept. In particular, the purpose of these initial questions was to assess whether the interdisciplinary context explored in the application was comprehensible in itself, and whether the pedagogical approach stimulated curiosity and offered incentives for further exploration of the application. The next stage in the interview featured a hands-on exploration of the application in its then-current state, followed by questions on both the learning content and the perceived usability. The interview then concluded with a discussion on final reflections.

5.2 Summative Testing

Two summative studies were conducted to evaluate the final prototype version. Both studies incorporated two evaluation tools, that is, the System Usability Scale (SUS) questionnaire and a custom survey. The former is a standardised usability questionnaire designed for post-study scenarios, that is, after completion of a list of tasks in a test scenario. The questionnaire consists of ten items, each of which provides a five point Likert scale response option. It was selected as a research instrument because of its concise format and sufficient reliability for benchmark comparison purposes [32]. The latter was designed to gain a better understanding of certain aspects related to the content and use of the application and thus to substantiate (or qualify) the results of the previous formative usability study. Furthermore, it served as a tool for collecting demographic data.

Both studies took place in formal testing environments, that is, a lecture for non-beginner computer science students (n=28) and a research seminar for advanced students of educational sciences (n=4). A time frame of approx.

Module Title	Sound and Music Concepts	Code Concepts
Learning By Playing	Synthesizer, Oscillation, Sample, Modulation, Waveshaping, Timbre, Delays	Variable, Loop, Function, Function Body, Function Argument, Parameter, Callback
Sound and Waves	Sound wave, Oscillation, Sinusoid / Sine Wave, Oscillator	Variable, Variable Declaration and Variable Initialization, Object, Argument
Frequency and Pitch	Frequency, Hertz, Pitch, Concert Pitch	Variable Value, Data Types, Number, String, Method Chaining
Different Waveforms	Fundamental Frequency, Harmonicity, Overtones/ Partials, Primary Waveforms (Sine, Square, Triangle, Sawtooth)	Array, For Loop, Code Comments
Dynamics and Loudness	Dynamics, Loudness, Sound Power, Sound Intensity, Decibel	Introduction to Programming Best Practices

Table 1: Overview of learning modules provided in the `muco` application

one hour was allocated in both cases. Most participants in both test settings were in their twenties. Given their study background, all participants in the first test group had prior knowledge and experience in programming. In the second group, only one participant reported to have some basic understanding of programming concepts. Surprisingly, the number of participants who have had some form of formal music education was relatively high in both groups (39,3% and 50%, respectively). While gender balance was maintained in the second group, the ratio of female to male participants in the first group was 1:3.

At the beginning of each study, a short overview of the research context and study was given. This included a short illustration of the main structure of the `muco` application. Subsequently, a task list was presented, listing the tasks which participants were asked to perform prior to responding to the questionnaire and survey via Google Forms [7]. Before sending out the links to the application and Google Forms, informed consent was obtained with respect to the processing and analysis of the collected data. Participants were then given 30-45' for exploring the content and functionality of the application and for responding to the Google Forms. Depending on the respective field of study of each test group, the survey was subjected to slight modifications.

6. RESULTS

This section presents the results obtained by means of both formative and summative test procedures. Overall, the results of the formative studies, that is, the interviews conducted with domain experts from the musical domain, highlighted general curiosity and acclaim with respect to the interdisciplinary learning context, but revealed mixed responses with regards to the learning content. The results obtained by means of the two summative studies involving non-beginner computer science and educational sciences students revealed mixed results with regards to the overall learning and teaching concept and the learning content provided in the application. The user interface and experience was praised by study participants in both study formats. The following thematically groups and describes the results in further detail.

6.1 Response on Interdisciplinary Concept

In the formative setup, all interviewees expressed great curiosity about the interdisciplinary learning concept provided in the application. Despite the general enthusiasm, however, some misunderstanding on the intended use case and target user group of the application was perceived. However, this only concerned one participant.

In the summative setup, in the first group (formed of computer science students), the general concept of combining music and music related concepts with computer science and computational thinking in order to teach programming raised some doubts. Though most respondents praised the playfulness and the individual features of the application (the code sonification and the game mechanics, particularly), some respondents did question the exact use case of the learning application. In the second group, the students with a background in educational sciences were more welcoming of the general concept and particularly highlighted the *Try It Yourself* sections in each learning module and emphasised their importance in the context of learning new concepts. Participants in both groups praised the playful and creative approach and game mechanics featured in the application, with the code sorting tasks attracting the most acclaim.

6.2 Response on Learning Content

In the formative studies, the response on the structure and content of the learning modules was mixed, ranging from very positive to neutral to fairly negative attitudes. Two participants expressed strong interest and explicitly articulated they would want to continue using the application once it featured more learning modules. One participant reported feeling neutral towards the application, and the overall teaching concept. Of the remaining two partic-

[7] https://about.google/intl/forms/ (accessed 31-10-21)

ipants, one expressed having felt overwhelmed with the coding-related concepts, and therefore discouraged to use the application in an informal (i.e. self-directed) learning context. The other participant was equally unenthusiastic, but for the opposite reason, that is, the learning modules were perceived as too fragmented and incremental. It should be noted that this participant was the one having had prior programming knowledge.

The responses in the summative studies were likewise mixed, ranging from participants describing the content as advancing too slowly to test users complaining about the length and complexity of the modules. Both groups also included participants who explicitly praised the learning content, emphasising its step-by-step approach and the text-highlighting feature of important concepts.

6.3 Response on User Experience

In both the formative and summative tests, the responses on the user experience of the `muco` application were very positive, with many participants pointing out the appealing interface design. Despite these general positive remarks, however, a number of participants (over all study formats) expressed they would appreciate some form of visualisation as a learning aid for certain topics. In particular, sound waves were mentioned as useful and suitable means for visualisation purposes.

6.4 SUS Score

As mentioned earlier, the SUS was applied as a research instrument in the summative tests. For the first participant group (formed of computer science students), the overall SUS score obtained amounted to 70.8 (of 100). This is slightly above the general average score of 68 (at the 50th percentile), and can thus be considered 'acceptable' [36]. Note that for the second group (formed of students of educational sciences), the SUS score was not computed due to the small sample size (n=4). Fig. 3 presents the overall SUS score as broken down into its usability and learnability subscores for the first group.

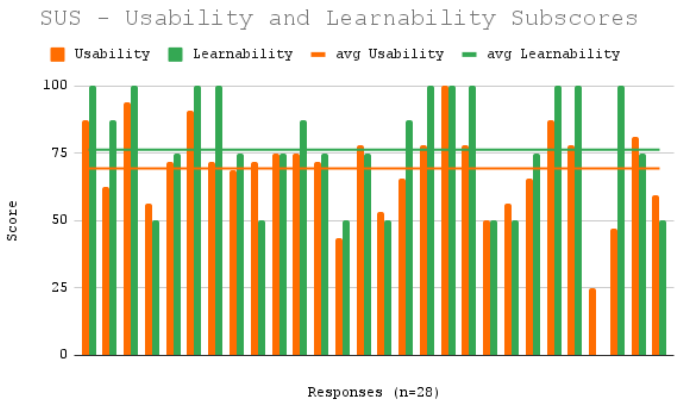


Figure 3: SUS subscores obtained in the formative study involving computer science students

6.5 Limitations

As with any other research method, expert interviews depict inherent weaknesses, not least due to their qualitative nature and the resulting risk of obtaining biased, subjective results. To address this shortcoming, the interviews conducted in the scope of the present work were complemented with two quantitative techniques, involving a standardised questionnaire and a broad-based survey. Despite these efforts, the total number of participants (that is, participants from all evaluation studies) remains rather small (N=37), thus limiting the significance of the results obtained.

Another limitation of this work concerns the characteristics of the test participants with respect to the following two aspects: First, regarding the knowledge and skill background of test participants, it must be emphasised that a large number of participants (i.e., those coming from a computer science study background) already had solid programming skills. Second, regarding the gender ratio, the gender imbalance encountered over all participants (across all evaluation studies) must be noted, with female participants accounting for only 27% of total participants.

On a final note, it should be emphasised that the results presented herein do not claim to be complete or exhaustive. Particularly, the concerns raised above necessitate the conduction of further usability studies with test groups which are both larger in size and more diverse with respect to both gender and each individual participants' background.

7. CONCLUSION

The present work focuses on the design and implementation of `muco`, a web-based music computing learning application which builds on the premise of combining music and music related concepts with computer science and computational thinking constructs to teach programming to beginners through coding music. Exploring and exploiting the potential of such interdisciplinary learning contexts is one of the principal considerations of STEAM education. Despite promising results of STEAM programs reported in the literature, researchers have likewise pointed to a number of integration challenges hindering large-scale implementation of STEAM-based curricula in school and classroom contexts [20]. In addressing these integration issues, the `muco` application was further conceptualised for use in non-formal (self-directed) learning contexts. To this end, the application leverages selected game mechanics to encourage learning persistence in such contexts [30].

Another issue commonly reported in the STEAM literature is that concerning the role and extent of arts integration [13]. Though not specifically addressed, in the course of conceptualising and defining the learning modules, a number of fundamental sound and music concepts were integrated were incorporated in an attempt to balance the roles of the disciplines considered.

The application was subjected to appropriate evaluation methods, involving both formative and summative usability tests depending on the level of maturity and functional scope of the respective development stage. Given the inter-

disciplinary nature of the learning concept featured in the application, test participants included domain experts from the fields of music, computer science and educational sciences. Formative testing was performed by means of expert interviews, whereas summative testing relied on the SUS standardised questionnaire and a custom survey as research tools. Summarising the results from both study formats, it can be said that the interdisciplinary learning context resulting from the combination of music and coding proves to be suitable for sparking student interest and motivation, and is therefore deemed worthwhile for further exploration. Beyond that, mixed responses were observed with regards to the structure, difficulty and appropriateness (for beginners) of the actual learning content provided by the `muco` application. This aligns with the results of other scholars who reported on similar interest increasing effects with STEAM education in a school teaching [12, 19].

The results of the studies conducted also revealed a (re)structuring of the learning content and a visualisation feature as potential areas for future work. Though the overall reaction on the structure and organisation of the learning content was mixed, dividing and restructuring the learning modules into smaller content parts would likely prove helpful and encouraging in the perceived learning experience. This would be especially true for younger audiences. Likewise, visualisations in the learning contexts can serve as a powerful tool for improving both understanding and memorisation of complex topics. Especially when faced with news concepts, visualisation can strongly accelerate and advance understanding and learning. For these reasons, the potential for visualisation features is subjected to further exploration in future iterations.

Apart from these content and design related issues, on a broader level, specific knowledge transfer potentials between the domains of music (including music theory, analysis and composition) and computing (including computational and algorithmic thinking, and programming) remain subject to more extensive empirical research.

Acknowledgments

The authors wish to thank the Center for Technology & Society (CTS) [8] for the funding received. The CTS is an inter-university research platform focusing on inter- and transdisciplinary development of technology, socio-technical research and inter-university cooperation.

8. REFERENCES

[1] H. K. Taube, "Computation and music," *Sound Musicianship: Understanding the Crafts of Music*, 2013.

[2] M. Müller, *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.

[3] S. Laato, S. Rauti, and E. Sutinen, "The role of music in 21st century education-comparing programming and music composing," in *2020 IEEE 20th International Conference on Advanced Learning Technologies (ICALT)*. IEEE, 2020, pp. 269–273.

[4] A. R. Brown, "Algorithms and computation in music education," in *The Oxford Handbook of Algorithmic Music*, 2018.

[5] N. Collins, *Origins of algorithmic thinking in music*, 01 2018, pp. 67–78.

[6] A. Misra, D. Blank, and D. Kumar, "A music context for teaching introductory computing," *ACM SIGCSE Bulletin*, vol. 41, no. 3, pp. 248–252, 2009.

[7] S. Aaron, A. F. Blackwell, and P. Burnard, "The development of sonic pi and its use in educational partnerships: Co-creating pedagogies for learning computer programming," *Journal of Music, Technology & Education*, vol. 9, no. 1, pp. 75–94, 2016.

[8] S. Siva, T. Im, T. McKlin, J. Freeman, and B. Magerko, "Using music to engage students in an introductory undergraduate programming course for non-majors," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 975–980. [Online]. Available: https://doi.org/10.1145/3159450.3159468

[9] V. Barr and C. Stephenson, "Bringing computational thinking to k-12: What is involved and what is the role of the computer science education community?" *ACM Inroads*, vol. 2, no. 1, p. 48–54, Feb. 2011. [Online]. Available: https://doi.org/10.1145/1929887.1929905

[10] F. J. García-Peñalvo, D. Reimann, and C. Maday, *Introducing Coding and Computational Thinking in the Schools: The TACCLE 3 – Coding Project Experience*. Cham: Springer International Publishing, 2018, pp. 213–226. [Online]. Available: https://doi.org/10.1007/978-3-319-93566-9_11

[11] P. Fotaris, T. Mastoras, R. Leinfellner, and Y. Rosunally, "Climbing up the leaderboard: An empirical study of applying gamification techniques to a computer programming class." *Electronic Journal of e-learning*, vol. 14, no. 2, pp. 94–110, 2016. [Online]. Available: https://eric.ed.gov/?id=EJ1101229

[12] F. Kayali, V. Schwarz, N. Luckner, and O. Hödl, "Play it again. digitale musikinstrumente im mintunterricht," *journal für lehrerInnenbildung jlb*, vol. 20, no. 1, pp. 54–66, 2020. [Online]. Available: https://elibrary.utb.de/doi/pdf/10.35468/jlb-01-2020_04

[13] E. Perignat and J. Katz-Buonincontro, "STEAM in practice and research: An integrative literature review," *Thinking Skills and Creativity*, vol. 31, pp. 31–43, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1871187118302190

[14] O. Shatunova, T. Anisimova, F. Sabirova, and O. Kalimullina, "STEAM as an innovative educational technology," *Journal of Social Studies Education Research*, vol. 10, no. 2, pp. 131–144, June 2019.

[8] https://cts.wien

[Online]. Available: https://www.learntechlib.org/p/216582

[15] G. Ozkan and U. U. Topsakal, "Investigating the effectiveness of STEAM education on students' conceptual understanding of force and energy topics," *Research in Science & Technological Education*, vol. 0, no. 0, pp. 1–20, 2020. [Online]. Available: https://doi.org/10.1080/02635143.2020.1769586

[16] M. Shamir, M. Kocherovsky, and C. Chung, "A paradigm for teaching math and computer science concepts in k-12 learning environment by integrating coding, animation, dance, music and art," in *2019 IEEE Integrated STEM Education Conference (ISEC)*, 2019, pp. 62–68.

[17] H. Thuneberg, H. Salmi, and F. Bogner, "How creativity, autonomy and visual reasoning contribute to cognitive learning in a STEAM hands-on inquiry-based math module," *Thinking Skills and Creativity*, vol. 29, pp. 153–160, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1871187118301159

[18] T. McKlin, B. Magerko, T. Lee, D. Wanzer, D. Edwards, and J. Freeman, "Authenticity and personal creativity: How earsketch affects student persistence," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 987–992. [Online]. Available: https://doi.org/10.1145/3159450.3159523

[19] A. Xambó Sedó, S. Saue, A. R. Jensenius, R. Støckert, and Ø. Brandtsegg, "NIME prototyping in teams: A participatory approach to teaching physical computing," in *Proceedings of the International Conference on New Interfaces for Musical Expression*. Universidade Federal do Rio Grande do Sul, 2019, pp. 216–221.

[20] D. Herro, C. Quigley, and H. Cian, "The challenges of STEAM instruction: Lessons from the field," *Action in Teacher Education*, vol. 41, no. 2, pp. 172–190, 2019. [Online]. Available: https://doi.org/10.1080/01626620.2018.1551159

[21] P. J. Denning, "The profession of it beyond computational thinking," *Commun. ACM*, vol. 52, no. 6, p. 28–30, Jun. 2009. [Online]. Available: https://doi-org.uaccess.univie.ac.at/10.1145/1516046.1516054

[22] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2012/08/Jeannette_Wing.pdf

[23] V. J. Shute, C. Sun, and J. Asbell-Clarke, "Demystifying computational thinking," *Educational Research Review*, vol. 22, pp. 142–158, 2017.

[24] T.-C. Hsu, S.-C. Chang, and Y.-T. Hung, "How to learn and how to teach computational thinking: Suggestions based on a review of the literature," *Computers & Education*, vol. 126, pp. 296–310, 2018. [Online]. Available: https://doi.org/10.1016/j.compedu.2018.07.004

[25] CSforALL, "CS for ALL computer science for all students," https://www.csforall.org/, 2016, accessed: 2021-07-30.

[26] Computer Science Teachers Association, "CSTA computer science teachers association," https://www.csteachers.org/, 2019, accessed: 2021-07-30.

[27] S. Kumar Basak, M. Wotto, and P. Belanger, "E-learning, m-learning and d-learning: Conceptual definition and comparative analysis," *E-Learning and Digital Media*, vol. 15, no. 4, pp. 191–216, 2018.

[28] K. M. Kapp, *The gamification of learning and instruction: game-based methods and strategies for training and education*. John Wiley & Sons, 2012.

[29] K. Squire and H. Jenkins, "Harnessing the power of games in education," *Insight*, vol. 3, no. 1, pp. 5–33, 2003.

[30] P. Mishra, C. Fahnoe, D. Henriksen, D.-P. R. Group *et al.*, "Creativity, self-directed learning and the architecture of technology rich environments," *TechTrends*, vol. 57, no. 1, pp. 10–13, 2013.

[31] J. Lazar, J. H. Feng, and H. Hochheiser, *Research methods in human-computer interaction*, 2nd ed. Morgan Kaufmann, 2017.

[32] J. Sauro and J. R. Lewis, *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann, 2016. [Online]. Available: https://www.perlego.com/book/1809426/quantifying-the-user-experience-pdf

[33] J. Nielsen, "Ten usability heuristics," 2005.

[34] B. Shneiderman, C. Plaisant, M. S. Cohen, S. Jacobs, N. Elmqvist, and N. Diakopoulos, *Designing the user interface: strategies for effective human-computer interaction*. Pearson, 2016.

[35] A. Blackwell, A. McLean, J. Noble, and J. Rohrhuber, "Collaboration and learning through live coding (Dagstuhl Seminar 13382)," *Dagstuhl Reports*, vol. 3, no. 9, pp. 130–168, 2014. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2014/4420

[36] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *International Journal of Human–Computer Interaction*, vol. 24, no. 6, pp. 574–594, 2008. [Online]. Available: https://doi.org/10.1080/10447310802205776