Learning to Code Through Web Audio: A Team-Based Learning Approach*

ANNA XAMBÓ,[1] **ROBIN STØCKERT,**[2] **ALEXANDER REFSUM JENSENIUS,**[3] **AND SIGURD SAUE**[2]
(anna.xambo@dmu.ac.uk) (robin.stockert@ntnu.no) (a.r.jensenius@imv.uio.no) (sigurd.saue@spotics.no)

[1]*Music, Technology and Innovation - Institute for Sonic Creativity (MTI*[2]*), De Montfort University, Leicester, UK*
[2]*Department of Music, Norwegian University of Science and Technology, Trondheim, Norway*
[3]*RITMO, Department of Musicology, University of Oslo, Oslo, Norway*

In this article, we discuss the challenges and opportunities provided by teaching programming using web audio technologies and adopting a team-based learning (TBL) approach among a mix of co-located and remote students, mostly novices in programming. The course has been designed for cross-campus teaching and teamwork, in alignment with the two-city master's programme in which it has been delivered. We present the results and findings from: (1) students' feedback; (2) software complexity metrics; (3) students' blog posts; and (4) teacher's reflections. We found that the nature of web audio as a browser-based environment, coupled with the collaborative nature of the course, were suitable for improving the students' level of confidence about their ability in programming. This approach promoted the creation of group course projects of a certain level of complexity, based on the students' interests and programming levels. We discuss the challenges of this approach, such as supporting smooth cross-campus interactions and assuring students' pre-knowledge in web technologies (HTML, CSS, JavaScript) for an optimal experience. We conclude by envisioning the scalability of this course to other distributed and remote learning scenarios in academic and professional settings. This is in line with the foreseen future scenario of cross-site interaction mediated through code.

0 INTRODUCTION

Programming can be considered an essential component of technology literacy, which is one of the identified skills of the 21st century [1]. There are no written rules on how to teach programming, but there exist studies with recommended teaching techniques, especially looking at how novices learn to program [2]. Motivation is a relevant factor, which can be enhanced with more interdisciplinary ways of learning, such as through so-called STEAM-based education, i.e. combining science, technology, engineering, arts, and mathematics [3]. It has been reported that teaching programming based on STEAM principles promotes a higher level of creativity [4]. It can also attract the interest of underrepresented populations in computing fields [5], which can provide a more diverse environment.

The educational research project Student Active Learning in a Two Campus Organization (SALTO)[1] aims to promote cross-campus learning as a future scenario in education [6]. Novel teaching strategies are investigated, such as team-based learning (TBL) [7], active learning [8] and flipped classroom [9]. A new international master's programme is related to this educational scheme named Music, Communication and Technology (MCT),[2] which is a collaboration between the Norwegian University of Science and Technology (NTNU) in Trondheim and the University of Oslo (UiO). The combination of simultaneous co-located vs remote learning is mediated through high-quality and low-latency audiovisual communication technologies [10]. The students are recruited with a broad range of artistic, scientific, and technological backgrounds. In the MCT courses, students often work together in interdisciplinary teams between the two campuses.

This article builds on our previous research [11], where we focused on understanding how to teach programming to novices within a TBL context more effectively. Here, we focus instead on the nature of the collaboration in a TBL context and how web audio programming can be a

*This is the Author's Accepted Manuscript. This manuscript has been accepted for publication on 2 September 2020: Anna Xambó, Robin Støckert, Alexander Refsum Jensenius, and Sigurd Saue, "Learning to Code Through Web Audio: A Team-Based Learning Approach," *J. Audio Eng. Soc.*, vol. 68, no. 10, pp. 727–737, (2020 October.). Please refer to the published version here: http://www.aes.org/e-lib/browse.cfm?elib=20989.

[1] https://www.ntnu.edu/salto
[2] https://www.ntnu.edu/studies/mmct

useful and affordable strategy to support both co-located and distributed collaboration when learning to code. The research question that this article addresses is: *What are the challenges and opportunities of using web audio to teach programming to a mixed cohort of co-located and remote students, who are mostly novices in programming, by adopting a team-based learning approach?* We hypothesise that using web audio technologies in a TBL context is a suitable approach to learning to code in a cross-campus scenario. The contributions of this article include a follow-up data analysis from [11], which investigates software complexity and students' blog posts in addition to students' feedback and teacher's reflections. Overall, the results and findings indicate that the nature of web audio as a browser-based environment, and the collaborative nature of the course, were suitable for improving the students' level of confidence about their ability in programming. The students were satisfied with their final course group projects, which had a certain level of complexity, and were related to the students' interests and programming levels. To conclude, we discuss the scalability of this course to other distributed and remote learning scenarios.

1 RELATED WORK

Our previous research highlighted how collaborative music live coding, which refers to the combination of the music improvisation practice of live coding and collaborative programming, is a promising learning strategy of programming [12]. In a more recent study on teaching physical computing to mostly novices [13], we also found that programming is a difficult skill to learn in a short period, as opposed to prototyping.

Web audio technologies have been successfully used in programming courses based on STEAM principles. For example, it was used in a summer programming music camp [14] to teach creative coding concepts through music by using different web audio libraries and frameworks, such as Gibber [15] and Tone.js [16]. Other courses are more focused on a particular library or environment. The Quint.js is a JavaScript library that combines visuals and audio to create interactive audiovisual programs and was used to teach music technology concepts to fine arts students [17]. EarSketch is a learning environment and curriculum that promotes coding through music-making, and is reported to have broadened participation in computing and music, particularly women [5]. Codecircle is a browser-based system for learning creative coding that supports real-time graphics and sound [4].

In this article, we present our strategies for teaching an audio programming course to an interdisciplinary group of students who are mostly novices in programming. We used a similar approach to [14] of exposing the students to different libraries and frameworks related to web audio, with the final aim of creating their project. This was also inspired by [18], who recommended to expose students to as many computer music languages as possible as a foundation for their future careers.

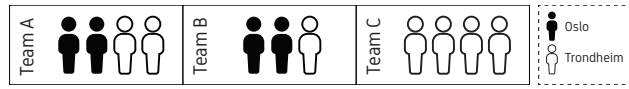


Fig. 1: The distribution of teams across the two campuses.

2 THE COURSE

2.1 Context

The MCT4048 Audio Programming course is an elective master course run in the Spring semesters. The aim is to provide a solid foundation in digital signal processing and audio-based application development. The integration of relevant technologies and platforms plays an important part to develop user-ready applications. There were several reasons for choosing web audio technologies for this course, namely:

- It is written in JavaScript, a modern programming language.
- It showcases the fundamental concepts of audio programming.
- It is relatively fast to get prototypes built.
- It can be widely distributed and freely accessed.
- It gives room for artistic expression.
- It is an employable skill.

2.2 Students

For the course edition analysed in this article, the number of students was 11 between the two campuses (2 women and 9 men), with 4 students in Oslo and 7 students in Trondheim. The group was international with students from Europe and Asia. English is the language required in the master's programme, so all students were fluent in English. Their backgrounds in programming were varied, predominantly novices. Three teams (*Teams A–C*) were formed during the second part of the course to create a final group project (see Figure 1). Two of the teams were cross-campus while one group was co-located. It is beyond the scope of this article to analyse the differences between both conditions. We focus instead on discussing a teaching approach that suits both scenarios, where it is assumed that a cross-campus scenario is more demanding compared to a co-located scenario.

2.3 Curriculum

We proposed a course model that works with the assumption that students have basic knowledge of web technologies (HTML, CSS, JavaScript) and some background in music (performance, production, theory). The course design also assumed that there is enough time for the students to develop both individual and group programming skills.

The course combined lectures and hands-on activities.[3] The lectures provided an overview of the fundamental concepts of audio programming. The hands-on workshops were based on building web applications using web audio tech-

[3] Slides and code are available online at: `https://github.com/axambo/audio-programming-workshop`

Fig. 2: Illustrations of the final student projects in the course. From left to right: (a) Touch the Alien, an interactive synthesizer, (b) Magic piano, a piano for children, and (c) Convolverizer, live processing of sound (artwork by Shreejay Shrestha).

nologies, both individually and in teams. The assessment of the course included the daily activity of the students and the results of two mini-projects that incorporated theory and practice seen in class.

Since it is part of a research-based master's programme, this course combined research-based activities with development activities. The course spanned 7 hours per day for 8 days (56 hours in 2 weeks). Each session began with setting up the students' computers with the tools for the tutorial of the day.

The first week was named "The Fundamentals," during which the students were asked to develop an individual mini-project. This allowed each student to work individually, familiarise themselves with the programming environment and find their way of expression (as if they were learning to play a musical instrument). The second week was named "The Extensions," during which the students were asked to develop a group mini-project. The students were exposed to additional features that could be useful to develop a more complex and complete prototype. It was expected that during the second week, the students should be more fluent in programming. The allocated class time for the mini-projects was 9.5 hours for the individual mini-project and 12.5 hours for the group mini-project. The general outline of the course was as follows:

- The Fundamentals & Individual Project (Week 1):
  - **Introduction Day**: Paper discussion about languages for computer music [19]. Discussion about the different languages and why to use the Web Audio API. Discussion and exercise about pseudocode.
  - **Day 1**: Warm-up discussion activity about the pros and cons of Web Audio (the outcome can be found in our previous related paper in the form of a mind map [11]). Tutorial of playing sounds. Individual mini-project development (3.5 hours).
  - **Day 2**: Warm-up discussion activity on an item learned the previous day in class. Tutorial of dealing with time using Tone.js [16]. Individual mini-project development (2 hours). Speedy presentations of the projects.
  - **Day 3**: Web Audio Conference (WAC) paper presentations part 1 (first half of the group, one paper per student). Tutorial of dealing with sound effects. Individual mini-project development (2 hours). Speedy presentations of the projects.
  - **Day 4**: WAC paper presentations part 2 (second half of the group, one paper per student). Tutorial of graphical user interfaces using NexusUI.js [20]. Individual mini-project development (2 hours). Final project presentations. Personal blog post writing on the project.
- The Extensions & Group Project (Week 2):
  - **Day 5**: Recap quiz of week 1. Tutorial on interactivity using Web MIDI API.[4] Group mini-project development (2.5 hours).
  - **Day 6**: Warm-up discussion activity on live coding. Tutorial of live coding using Gibber [15]. Group mini-project development (2 hours). Speedy presentations of the projects.
  - **Day 7**: Tutorial of mobile music and responsive design. Group mini-project development (4 hours). Speedy presentations of the projects.
  - **Day 8**: Tutorial of AudioWorklets [21]. Group mini-project development (4 hours). Final project presentations. Group blog post writing with reflections on the project.

At the beginning of the course, the teacher and students decided to work with the same code editor (Visual Studio Code) and web browser (Chrome). This facilitated that all students could follow the hands-on tutorials and could debug in collaboration or show their problems to the teacher or group if needed.

3 THE FINAL STUDENTS' PROJECTS

During the first week, the students developed a total number of 10 individual mini-projects, whilst in the second week, the students developed three group mini-projects. A total number of 13 blog posts about each mini-project were written. The blog posts were delivered as assignments with a suggested structure to follow (see Section 4.3). In most cases, the students explained how the system worked tech-

[4] http://webaudio.github.io/web-midi-api

nically and conceptually. They often also shared their code via a code repository and demonstrated how the system worked with embedded videos or live demos. Next, a brief description of the three group mini-projects is provided.

3.1 *Touch the Alien*

As shown in Figure 2a, this project is a web audio synthesiser offering touchscreen functionality suitable for a smartphone or tablet. The audio engine is a combination of oscillators and effects (e.g. delay, phaser, chorus). It is possible to control the filters via an interactive canvas, combined with buttons and sliders. The technologies used include Javascript with the Web Audio API, CSS, HTML5, and the audio effects library Tuna.[5] The aim was to combine visuals with sounds based on the work that the team members did during their individual mini-projects in the previous week. In addition to Visual Studio Code as their code editor, GitHub was used to share their code.

3.2 *The Magic Piano*

This project is based on a piano for children that plays the right notes of the chosen melody irrespective of whether the player hits the right or wrong key (Figure 2b). The prototype was inspired by one student who wanted to design a 'magic' piano for his daughter. The technologies used include Web MIDI API, NexusUI.js, Tone.js, JSON, CSS and a MIDI keyboard. Currently, the prototype supports two songs: *ABCD* and *Alle fugler*. Visual Studio Live Share was used to view and discuss the same code in real time, GitHub was used for sharing the code among them when they were working offline, and Zoom was used to communicate and share their screens.

3.3 *Convolverizer*

This project was built around real-time processing of ambient sound, voice or live instruments using a convolution effect based on a pre-loaded sound file (e.g. a drum solo). The prototype has a toggle button and slider which provides the user with control of the application, whilst a canvas is used for live visualisation (see Figure 2c). The technologies used include P5.js,[6] an audio interface, a guitar, and a microphone. The motivation was to create a modular solution for live performance, suitable for different string instruments (e.g. guitar, bass guitar) related to the team members' music interests. Apart from Visual Studio Code, the working tools included Zoom and Discord.

4 RESEARCH METHODS

Based on our research question, we analysed: (1) students' feedback; (2) software complexity metrics; (3) students' blog posts; and (4) teacher's reflections.

[5] https://github.com/Theodeus/tuna

[6] https://p5js.org

4.1 Students' Feedback Questionnaire

Adapted from our previous physical computing course [13], we distributed a voluntary pre-questionnaire and post-questionnaire with the same 5-point Likert-item questions to the students. The questions ranged from asking the level of confidence (*1 = not at all confident; 2 = a little confident; 3 = somewhat confident; 4 = highly confident; 5 = extremely confident*) about their ability for programming (Q1), computational thinking (Q2), prototyping (Q3), instrument building (Q4), reflective practice (Q5), teamworking (Q6), and individual working (Q7). There were also questions about the level of agreement (*1 = strongly disagree; 2 = disagree; 3 = neutral; 4 = agree; 5 = strongly agree*) with a set of statements on their intention to continue courses related to STEM fields (Q8), to continue their education in STEM fields (Q9), and to use their STEM knowledge in their future careers (Q10). The students were also asked about their level of agreement of the extent to which they can understand the purpose of audio programming (Q11), can describe the process of programming an interactive musical prototype (Q12), and can apply the technique of programming an interactive musical prototype to their work (Q13). The post-questionnaire also included three open questions about what the students liked best and least about the course, and how the course could be improved for future editions.

The data analysis was done using R.[7] Bar plots for the Likert items were produced (*likert*[8] package). A two-tailed Wilcoxon signed-rank test [22] was conducted to examine whether there was a significant difference between the pre- and post-questionnaires (R *stats* package and *coin*[9] package) as well as the Wilcoxon effect size (*rstatix*[10] package). The open questions were analysed with a bottom-up thematic analysis [23] approach to identify common themes.

4.2 Software Complexity Metrics

To analyse the overall level of complexity of the code developed by the students, we used established software complexity metrics. This included JavaScript implementations of Halstead metrics [24] and the cyclomatic complexity metric defined by Thomas McCabe [25]. The JavaScript implementations used were *plato*,[11] *halstead-metrics-cli*,[12] and *complexity-report*.[13]

Software complexity metrics are traditionally used to help programmers to improve the efficiency of their code or to prevent plagiarism in the classroom [26]. Here we used it to identify whether there is any general difference between the individual and group projects in terms of complexity of the code. We acknowledge that complexity is not the same as understandability or learnability, and therefore we combine these results with the accounts of the students' blog

[7] https://www.r-project.org

[8] https://cran.r-project.org/web/packages/likert

[9] https://cran.r-project.org/web/packages/coin

[10] https://cran.r-project.org/web/packages/rstatix

[11] https://github.com/es-analysis/plato

[12] https://www.npmjs.com/package/halstead-metrics-cli

[13] https://github.com/escomplex/complexity-report

posts to characterise the quality of the team-based learning experience. We also recognise that, by course design, it is expected a better work delivered in the second week.

4.3 Students' Blog Posts

The writing of blog posts, and consequent reflections on their individual and group projects, were mandatory activities in the course. The students were provided with a suggested structure for the blog post: description, timeline, division of labour, challenges, achievements, and future work. It was also recommended to add links to the source code and a live demo. Publishing blog posts during the course was helpful not only for the students; it also worked as a tool for the teacher to address any issues and enquiries while running the course.

We analysed the three group blog posts with a top-down thematic analysis [23] perspective using the NVivo software.[14] Here, we mainly looked into how the students reported about: (1) collaboration strategies and tools; (2) division of labour; (3) individual vs group learning; (4) challenges; and (5) achievements.

4.4 Teacher's Reflections

The reflections of the teacher (the first author) were investigated using a bottom-up approach to thematic analysis [23]. The reflections included notes about the realisation of the course: emerging issues discussed, relevant decisions taken, and lessons learned.

5 RESULTS

5.1 Students' Feedback

A total of 11 students responded to the pre- and post-questionnaires, out of which 9 students responded to the paired questionnaire (n=9). For the general questions about the course, the 10 students who replied the post-questionnaire (referred to as U1–U10) reported that they liked best:

- **Learning content** (4 occurrences): *"broadening the perspective"* (U8); *"learning new libraries and frameworks"* (U4); *"learning the basic building blocks of the Web Audio API"* (U2); *"learning live coding"* (U2).
- **Learning process and course outcomes** (3 occurrences): *"more security and confidence in programming"* (U3); *"build applications"* (U1); *"collaborative working"* (U5).
- **Course design** (4 occurrences): *"freedom for creativity"* (U7); *"work hands-on on the code and learn it"* (U10); *"the combination of lessons and hands-on practice and prototyping"* (U9); *"fun to learn how to code"* (U6).

The students liked least:

- **Be a novice programmer** (4 occurrences): *"lack of basic programming skills"* (U8); *"not knowing how to program"* (U3); *"individual working was hard without asking permanently for help"* (U5); *"as a beginner the need of basic content for a longer period of time"* (U9).
- **Intensive course format and remote programming in teams** (4 occurrences): *"too short and condensed"* (U1); *"a bit too fast and packed"* (U10); *"only 2 weeks in the whole semester seems to be a pity!"* (U9); *"hard to collaborate with coding over distances"* (U6).
- **Research-based web programming course** (2 occurrences): *"added complexity of working with other web technologies, which can take a bit of focus away from audio programming"* (U2); *"too much focus on other things than programming (blog, presentation)"* (U4).

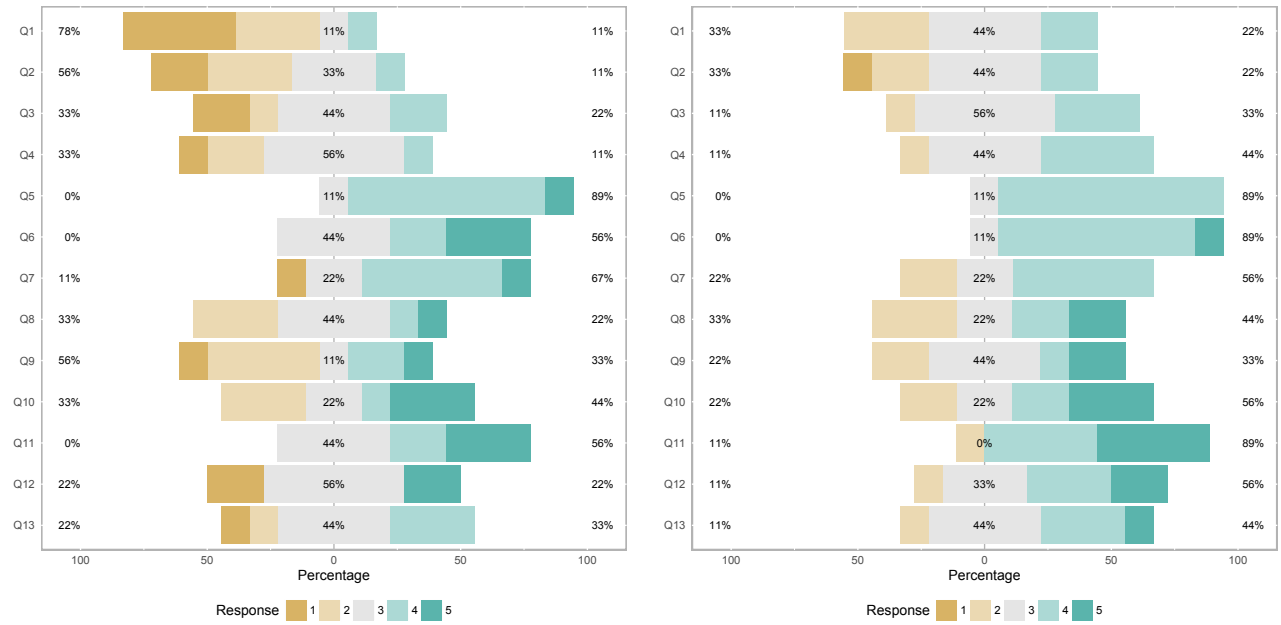
The students suggested several aspects to be improved in future runs of the course, including:

- **Avoid intensive course format** (4 occurrences): *"making it less intense and more spread during the term"* (U8); *"extend to more than 2 weeks workshop"* (U1); *"adding another week and slowing down the process will help for absolute beginners a lot"* (U10); *"we could have learnt more if the course had run for a longer period of time"* (U9).
- **Request pre-knowledge in programming and web technologies** (3 occurrences): *"having a basic level of training in the beginning"* (U8); *"required pre-knowledge in programming from all students"* (U4); *"an introductory lecture or two in the absolute basics in coding"* (U7).
- **Satisfy both novices and experts** (3 occurrences): *"the course was well taught, even though the level was very high, more time [was] needed to evaluate"* (U3); *"the way it was this year would suit more for students with prior understanding of programming to some level"* (U9); *"define more clearly incremental tasks related to the curriculum, which should also have the possibility of extra challenges for those who are on a higher level"* (U2).

Figure 3 shows the percentages of the level of confidence and agreement, which tended to be more positive in the post-questionnaire (Mdn=4, M=3.47) than in the pre-questionnaire (Mdn=3, M=3.09). These results align with the results from our previous course in physical computing [13], which indicate that the pedagogical techniques are perceived positively.

We acknowledge that our statistical analysis is based on a small sample, and therefore the results should be considered as a general indicator. The results of the two-tailed Wilcoxon signed-rank test were only significant in Q1 about the students' level of confidence of programming: V=0, z=−2.7136, p<.05, where a large effect size was detected, r=0.917. This indicates that the median of the

[14] https://www.qsrinternational.com/nvivo



(a) Bar plot of the pre-questionnaire responses.

(b) Bar plot of the post-questionnaire responses.

Fig. 3: Bar plot for the results of thirteen (Q1–Q13) 5-point Likert-item questions (n=9). Questions: Q1 programming; Q2 computational thinking; Q3 prototyping; Q4 instrument building; Q5 reflective practice; Q6 teamworking; Q7 individual working; Q8 continue STEM courses; Q9 continue STEM education; Q10 future use of STEM knowledge; Q11 understanding of audio programming; Q12 understanding of programming interactive musical prototypes; and Q13 programming interactive musical prototypes. The detailed questions can be found in Section 4.1.

post-questionnaire (Mdn=3) was significantly higher than the median of the pre-questionnaire (Mdn=2). This contrasts with the level of confidence of programming achieved in the physical computing course, which was one of the less developed skills.

For the rest of the questions, no significant difference was found. Although the level of confidence of prototyping (Q3) and instrument building (Q4) slightly improved, the difference was not significant. The focus of the course was on audio programming, so, understandably, these other desired skills remained secondary. Similarly, the intention to continue STEM courses (Q8) and STEM education (Q9) improved slightly, but with no significant difference, which points to the need of more time to stimulate students' interest into STEM careers. The understanding of audio programming (Q11) polarised a little bit more in the post-questionnaire, probably associated with the need of learning additional tools related to web development (as discussed earlier in the open questions). The level of confidence of teamworking (Q6) and reflective practice (Q5) slightly increased from an already high score, two aspects that are explicitly promoted across the different courses in the MCT program. The level of confidence of individual work (Q7) changed from extreme to moderate positive and negative opinions, yet with no statistical difference. Individual work was an essential component of this course during the first week, and it seems to be valued by the students, but it needs to be better integrated so that both experts and novices acknowledge an improvement.

5.2 Code Complexity

Table 1 compares the code complexity metrics at an individual vs group level, sorted by teams. Before detailing the code complexity analysis, it is worth describing how each team compiled the code for the final group project. Team A decided to divide the work by functionality so that each team member developed their own piece of code. This was merged by one of the students on the last day by keeping the code in separate JavaScript files and tweaking as necessary. Team B collaboratively worked on a GitHub repository where it is reported that the three team members contributed with 31 commits, 19 commits and 13 commits, respectively. This team also reported a division of labour by functionality, yet collaborative coding and troubleshooting were highlighted as common tasks. Team C approached the project with a collaborative coding style, where the work was divided into sub-tasks. The code was uploaded on a GitHub repository by one of the team members, yet it does not identify code authorship.

Regarding code complexity, Team A had consistent progress from individual work in the first week to group work in the second week, where data is excluded from the fourth student of this team who could not contribute to the two projects. Two of the three students were already familiar with web technologies. The group's approach was additive, where the final project had 515 lines of code in total (almost the double of each of the individual projects). The average maintainability of the code slightly evened out to 66.5, which indicates that the code from individual and

Table 1: Code complexity metrics by teams

Team	Lines of Code	Maintainability	Cyclomatic Complexity	Difficulty	Time Required to Program (h)	# Delivered Bugs
A	515	66.5	22	125.77	29.84	5.17
A individual *	*Mdn*=220	*Mdn*=72.4	*Mdn*=5	*Mdn*=108.01	*Mdn*=14.07	*Mdn*=3.13
	M=206.67, *SD*=88.75	*M*=69.13, *SD*=6.74	*M*=12, *SD*=13.89	*M*=95.29, *SD*=35.88	*M*=11.13, *SD*=6.4	*M*=2.6, *SD*=0.91
B	255	60.91	22	105.3	16.83	3.53
B individual	*Mdn*=209	*Mdn*=68.5	*Mdn*=27	*Mdn*=101.7	*Mdn*=9.83	*Mdn*=2.47
	M=306, *SD*=214.62	*M*=63.29, *SD*=9.68	*M*=24.33, *SD*=12.22	*M*=148, *SD*=116.79	*M*=45.23, *SD*=64.63	*M*=5.77, *SD*=6.33
C	108	64.95	6	37.24	1.71	0.77
C individual **	*Mdn*=136	*Mdn*=67.46	*Mdn*=3	*Mdn*=71.72	*Mdn*=5.6	*Mdn*=1.7
	M=197, *SD*=110.01	*M*=60.58, *SD*=12.1	*M*=18.67, *SD*=28.01	*M*=94.77, *SD*=58.34	*M*=16.47, *SD*=20.79	*M*=3.08, *SD*=2.83

* Data is excluded from a student of this team who could not contribute to the individual and group projects.
** Data is excluded from a student of this team who could not contribute to the individual project.

group work has good maintainability. The complexity of the program measured as cyclomatic complexity evened out to 22.0, which indicates that there has been an improvement in reducing the complexity of the code and making it more maintainable. The difficulty of the program rated the maximum value of 125.8, which is expected by the course design. The estimation of the time required to program in hours raised to 29.8 hours. This aligns with the increased number of lines of code and is also expected. The number of delivered bugs or errors almost doubled to 5.2, which is not expected. This points to a lack of time in debugging the final code, as it was merged on the last day (see Section 6.1 for further details).

Team B evened out the code complexity results (e.g. lines of code, average maintainability, cyclomatic complexity, program difficulty, number of delivered bugs) yielded from one more experienced programmer and two novices. This indicates that all team members contributed to the final project, irrespective of their programming level. These results suggest that the code from individual and group work has good maintainability, there has been an improvement in reducing the complexity of the code and making it more maintainable, all the team members contributed and collaborated as expected from a course design viewpoint, and there was a general reduction of bugs. The benefits for the novices were a clear improvement in their code. For the expert programmer the benefits were three-fold: a reduction of the number of bugs (due to probably more group testing), an even distribution of the work (also evidenced in the code repository), and an improvement of the maintainability of the code.

Team C was the only group with all members being programming novices. In the individual projects, one of the students borrowed the code from one of the most experienced students in the class, which can be seen as an outlier in the individual results. The individual results should, therefore, be considered with a pinch of salt. As for the final project, this team preferred to use an easy to use, pre-built library (P5.js), which was covered in a previous MCT course [13]. The code complexity result is relatively low as the effort of coding is little compared to what was expected. The time required to program resulted in 1.7 hours, although the group worked as hard as the others during class time. As we will discuss later, the team succeeded in completing the final project, but they virtually 'skipped' lower-level programming, which is reflected in the code complexity score. This team was more concerned with having something working using a high-level approach to programming than in developing complex code. This confirms that, for an optimal experience, this course should be taken by students with pre-knowledge in web technologies so that the students can focus more on the task of audio programming. Teaming novices with experts should also help, but this is not always feasible.

6 FINDINGS

6.1 Students' Blog Posts

We found the following number of occurrences for each team and non-exclusive theme: **collaboration strategies and tools (T1)** (17 occurrences: 7(A), 3(B), 7(C)); **division of labour (T2)** (16 occurrences: 7(A), 4(B), 5(C)); **individual vs group learning (T3)** (9 occurrences: 6(A), 1(B), 2(C)); **challenges (T4)** (17 occurrences: 3(A), 5(B), 9(C)); and **achievements (T5)** (11 occurrences: 7(A), 1(B), 3(C)). Next, we summarise key points illustrated with exemplary quotes, classified with the group name, theme number, and reference number, e.g. *"A-T1-Ref3"*.

Team A decided to divide the work into smaller pieces to be developed individually, and merged the code at the end: *"On the last day, we finished the prototype by combining all codes together and made final touches on design and functions. This meant tidying up some of our code, bug fixing and making each intended function work properly." (A-T1-Ref5).* The team acknowledged as an achievement *"to combine our three branches of code. (. . .) It showed that our organisation of labour worked out in the end!" (A-T5-Ref6).* The team recognised that *"working on each other's code together was certainly both challenging and rewarding" (A-T4-Ref2)*, but overall admitted as a success that *"for the workshop we materialised a working prototype!" (A-T5-Ref1).*

Team B worked continuously in collaboration, both synchronously with live coding and asynchronously with a code

repository: *"We worked in a collaborative way where we shared screens with each other and worked on the same document and files. We set up the VS [Visual Studio] Live Share to view the same code in real-time and to discuss the code. We used GitHub for sharing the code among us when we worked offline. We used Zoom to communicate and share screens." (B-T1-Ref2).* Overall, the team recognised the challenge and achievement of working in a group: *"Collaborative coding was a great challenge, but we feel like we managed to have a good workflow as a group. Our idea for making an educational tool for beginners to learn playing a simple melody on the piano, has developed into a prototype that we are proud of." (B-T5-Ref1).*

Team C also worked continuously in collaboration, but preferred not to divide the work into main roles: *"Since the level of programming expertise was more or less equally low distributed throughout the group, we left the division of roles open." (C-T1-Ref2).* The four members of this group were on the same location: *"We established a main hub in one of the group rooms and put the code from Visual Studio Code on the screen. From there, we brainstormed and prototyped together." (C-T1-Ref5).* The team acknowledged that *"[ending] up with a working prototype after this course is very pleasing." (C-T5-Ref1)*, but also recognised the struggle with their prior knowledge with programming: *"There was a bit of frustration tied with the difficulty of the task we had at hand, but through that, we managed to have a good working relationship and good teamwork." (C-T1-Ref7, C-T4-Ref7).*

6.2 Teacher's Reflections

The three group mini-projects were successful in completion. Although varied in theme, there are some similarities. First, the teams built on the code developed individually during the previous week, and they all faced challenges with merging their code and working with collaborative coding approaches. This was less of a problem for Team C, who built on code from an existing library seen in a previous MCT course [13] for the group project. Second, they all used a range of web audio and web technologies, some of them seen in class, but some of them explored autonomously or in previous courses. This is an excellent example of how the spirit of prototyping helps in finding the best tools and combining them to convey a project idea. Finally, they all combined software with hardware.

As discussed in our previous paper [11], we identified five prominent themes, of which we could mention: (1) **individual vs group work**: promoting individual work before group work facilitates the development of personal expression using code; (2) **shared coding experiences**: using the same working tools (e.g. code editor, browser) throughout the course ensures there is a common language and understanding irrespective of the student's programming level; promoting the use of real-time synchronous collaborative tools (e.g. collaborative live coding features) and practices (e.g. sharing the screen either peer-to-peer, peer-to-group, and group-to-group) allows for smooth interactions irrespective of the physical distances among collaborators; and encouraging the use of asynchronous collaborative tools (e.g. shared repositories) helps to keep the momentum beyond the course time as well as documents the process of collaboration; and (3) **web audio vs web technologies**: making sure that the students have the required pre-knowledge in web technologies in order to balance better novices vs experts should be taken rigorously.

7 DISCUSSION

Overall, learning how to program is a slow endeavour [2], but we have seen that learning to code through web audio—through a TBL methodology—can be a suitable approach for both cross-campus (Teams A and B) and co-located (Team C) groups. We found that the nature of web audio as a browser-based environment and the collaborative nature of the course were suitable for improving the students' level of confidence about their ability in programming, both in co-located and remote scenarios. The course design of starting with individual work and then move on to cultivate group work was also suitable for promoting the creation of final projects with a certain level of complexity. This also helped to relate the projects to the students' interests, in alignment with the principles of STEAM education [3].

On the challenging side, we identified the importance of supporting smooth cross-campus interactions in both the classroom and for the teams' workspaces. It is still an open question of what are the best tools that can be used to provide a satisfactory experience for remote collaborations mediated by code. The teacher and students of this course were able to explore various options, but more research is needed to find solutions to different demands. This should be based on each student cohort, but also each team's workflow and dynamics. Another open question is how to ensure that the students have the required pre-knowledge and skills with web technologies (HTML, CSS, JavaScript) before the start of the course. Such pre-knowledge is crucial, since it requires some time to develop, and is typically out of the scope of the programme's curriculum. Prolonging the learning experience is another challenge and an important one to tackle to improve the students' interest in continuing in STEM fields. This requires further development of the curriculum (curated resources, meetups, and so on) and better integration with TBL techniques. One approach here could be the Readiness Assurance Process (RAP) [7]. Finally, it would be relevant to assess further the students' level of self-confidence from a self-efficacy perspective in order to identify the strength of their belief in their ability in programming [27].

We envision that this course could be further developed into a completely online activity, with students across the globe. In this new scenario, we could replace the two physical classrooms used in the course with a virtual classroom (e.g. a collaborative virtual learning environment), in which dividing the students into smaller remote working groups is possible. For the moment, we recommend to keep this course at a scale of 10–20 students and create teams of 4–5 members, so that the current content and learning quality experience keeps similar to the course discussed here. We

recommend encouraging students to use a shared tool for online reflection (e.g. blogging) as a core part to reflect on their learning process. The assessment of the course should also be scalable using the same research methods presented here, yet we foresee the need for more coders for the thematic analysis, where intercoder agreement [28] would be useful. We also envision that distributed and remote professional settings might get inspired and use web audio for prototyping audio programming ideas involving mixed groups of team members, with different levels of programming.

8 CONCLUSION

In this article, we presented an audio programming course using web audio technologies targeted at an interdisciplinary group of master students who are mostly novices in programming. The collected data (including students' feedback, software complexity metrics, students' blog posts and teacher's reflections) indicated that web audio technologies using a TBL approach is a suitable programming learning approach. However, web technologies (HTML, CSS, JavaScript), and therefore basic programming concepts, should be requested as prior knowledge for an optimal experience. The final group projects showed the potential of addressing complexity with creativity and collaboration, which was partly possible due to the course design of first starting with individual work. It is still challenging to teach programming across two campuses, but applying techniques from collaborative live coding (e.g. sharing the screen and the code editor) can positively counterbalance the challenge.

In the future, we are interested in investigating whether web audio works better for TBL activities compared to native sound/music programming environments (e.g. Max/MSP or SuperCollider). From a STEAM education perspective, we also plan to deliver this course to students who are not necessarily musicians. This can then be an entry point not only to programming but also to computer music and music software. We also hope to explore further the potential of our approach but applied to distance learning, which is a follow-up of our exploration of more sustainable forms of education.

9 ACKNOWLEDGMENTS

The authors wish to thank the students who participated in the course and to the reviewers of the manuscript for their valuable suggestions. The authors are thankful to the jury members of the WAC 2019 Best Paper Award, Jan Monschke, Garth Paine, and Ariane Stolfi, for awarding our paper presented at WAC 2019, and to the WAC community for informal feedback. Also, thanks to the MCT teachers Daniel Formo, Anders Tveit and Kristian Nymoen for their technical help during the realisation of the course discussed in this article. This work was partially supported by the NTNU SALTO project (80340480). Most of the data collection and analysis of this research was carried out while the first author was at the Norwegian University of Science and Technology.

10 REFERENCES

[1] B. Pearlman, "Making 21st Century Schools: Creating Learner-Centered Schoolplaces/Workplaces for a New Culture of Students at Work," *J. Educ. Technol.*, pp. 14–19 (2009).

[2] A. Robins, J. Rountree, N. Rountree, "Learning and Teaching Programming: A Review and Discussion," *Comput. Sci. Educ.*, vol. 13, no. 2, pp. 137–172 (2003), doi: https://doi.org/10.1076/csed.13.2.137.14200.

[3] J. Maeda, "STEM + Art = STEAM," *The STEAM Journal*, vol. 1, no. 1, pp. 34:1–34:3 (2013), doi:https://doi.org/10.5642/steam.201301.34.

[4] M. Yee-King, M. Grierson, M. d'Inverno, "STEAM WORKS: Student Coders Experiment More and Experimenters Gain Higher Grades," presented at the *Proc. IEEE Global Eng. Educ. Conf.*, pp. 359–366 (2017), doi:https://doi.org/10.1109/educon.2017.7942873.

[5] B. Magerko, J. Freeman, T. Mcklin, M. Reilly, E. Livingston, S. Mccoid, A. Crews-Brown, "Earsketch: A STEAM-based Approach for Underrepresented Populations in High School Computer Science Education," *ACM Trans. Comput. Educ.*, vol. 16, no. 4, p. 14 (2016), doi: https://doi.org/10.1145/2886418.

[6] R. Støckert, A. R. Jensenius, S. Saue, "Framework for a Novel Two-Campus Master's Programme in Music, Communication and Technology Between the University of Oslo and the Norwegian University of Science and Technology in Trondheim," presented at the *Proc. Int. Conf. of Educ., Res. and Innov.*, pp. 5831–5840 (2017), doi: https://doi.org/10.21125/iceri.2017.1526.

[7] L. K. Michaelsen, A. B. Knight, L. D. Fink, *Team-based Learning: A Transformative Use of Small Groups in College Teaching* (Stylus Publishing, Sterling, VA) (2004).

[8] C. J. Ballen, C. Wieman, S. Salehi, J. B. Searle, "Enhancing Diversity in Undergraduate Science: Self-Efficacy Drives Performance Gains with Active Learning," *CBE—Life Sciences Education*, vol. 16, no. 4, pp. 1–6 (2017), doi:https://doi.org/10.1187/cbe.16-12-0344.

[9] J. Bergmann, A. Sams, *Flip Your Classroom: Reach Every Student in Every Class Every Day* (The Association for Supervision and Curriculum Development, Alexandria, VA) (2012), doi:https://doi.org/10.1111/teth.12165.

[10] R. Støckert, G. A. Stoica, "Finding the Right Pedagogy and Related Prerequisites for A Two-Campus Learning Environment," presented at the *Proc. Int. eLearn. & Softw. for Educ.*, pp. 219–228 (2018), doi:https://doi.org/10.12753/2066-026X-18-030.

[11] A. Xambó, R. Støckert, A. R. Jensenius, S. Saue, "Facilitating Team-Based Programming Learning with Web Audio," presented at the *Proc. of the Web Audio Conf. 2019*, pp. 2–7 (2019).

[12] A. Xambó, G. Roma, P. Shah, T. Tsuchiya, J. Freeman, B. Magerko, "Turn-taking and Online Chatting in Co-located and Remote Collaborative Music Live Coding," *J. Audio Eng. Soc.*, vol. 66, no. 4, pp. 253–266 (2018), doi: https://doi.org/10.17743/jaes.2018.0024.

[13] A. Xambó, S. Saue, A. R. Jensenius, R. Støckert, Ø. Brandtsegg, "NIME Prototyping in Teams: A Participa-

tory Approach to Teaching Physical Computing," presented at the *Proc. New Interfaces for Musical Expression*, pp. 216–221 (2019), doi:https://doi.org/10.5281/zenodo.3672932.

[14] J. Allison, D. Holmes, Z. Berkowitz, A. Pfalz, W. Conlin, N. Hwang, B. Taylor, "Programming Music Camp: Using Web Audio to Teach Creative Coding," presented at the *Proc. Web Audio Conf.* (2016).

[15] C. Roberts, J. Kuchera-Morin, "Gibber: Live Coding Audio in the Browser," presented at the *Proc. Int. Computer Music Conf.* (2012).

[16] Y. Mann, "Interactive Music with Tone.js," presented at the *Proc. Web Audio Conf.* (2015).

[17] I. G. Burleigh, T. Schaller, "Quint.js: A JavaScript Library for Teaching Music Technology to Fine Arts Students," presented at the *Proc. Web Audio Conf.* (2015).

[18] A. Pošćić, G. Kreković, "Ecosystems of Visual Programming Languages for Music Creation: A Quantitative Study," *J. Audio Eng. Soc.*, vol. 66, no. 6, pp. 486–494 (2018), doi:https://doi.org/10.17743/jaes.2018.0028.

[19] R. B. Dannenberg, "Languages for Computer Music," *Frontiers in Digital Humanities*, vol. 5, pp. 26:1–26:13 (2018), doi:https://doi.org/10.3389/fdigh.2018.00026.

[20] B. Taylor, J. T. Allison, W. Conlin, Y. Oh, D. Holmes, "Simplified Expressive Mobile Development with NexusUI, NexusUp, and NexusDrop," presented at the *Proc. New Interfaces for Musical Expression*, pp. 257–262 (2014), doi:https://doi.org/10.5281/zenodo.1178951.

[21] H. Choi, "AudioWorklet: The Future of Web Audio," presented at the *Proc. Int. Computer Music Conf.* (2018).

[22] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83 (1945), doi:https://doi.org/10.2307/3001968.

[23] V. Braun, V. Clarke, "Using Thematic Analysis in Psychology," *Qual. Res. Psychol.*, vol. 3, no. 2, pp. 77–101 (2006), doi:https://doi.org/10.1191/1478088706qp063oa.

[24] M. H. Halstead, *et al.*, *Elements of Software Science* (Elsevier Science Inc., New York, NY) (1977).

[25] T. J. McCabe, "A Complexity Measure," *IEEE T. on Software Eng.*, vol. SE-2, no. 4, pp. 308–320 (1976), doi:https://doi.org/10.1109/tse.1976.233837.

[26] R. J. Leach, "Using Metrics to Evaluate Student Programs," *SIGCSE Bull.*, vol. 27, no. 2, pp. 41–43 (1995), doi:https://doi.org/10.1145/201998.202010.

[27] A. Bandura, *Self-Efficacy: The Exercise of Control* (Worth Publishers, New York, NY) (1997).

[28] J. Saldaña, *The Coding Manual for Qualitative Researchers* (Sage, London), 2nd ed. (2013).

THE AUTHORS

Anna Xambó

Robin Støckert

Alexander Refsum Jensenius

Sigurd Saue

Anna Xambó is a Senior Lecturer in Music and Audio Technology at De Montfort University. Her research and practice focus on sound and music computing systems looking at novel approaches to collaborative, participatory, and live coding experiences. She has contributed to the Web Audio Conference (WAC) as a program committee member and author since 2016, and has served as music/artworks co-chair of WAC 2016, and general co-chair of WAC 2019. She has also a special interest in improving the representation of women in music technology.

●

Robin Støckert is an assistant professor at NTNU and an AV expert with over 35 years of experience in the design and creation of arenas for interaction, experimentation, collaboration and communication. He has been involved in several EU-projects relating to the design, construction, and use of future learning spaces and interactive tools in higher education. He is the project manager for the SALTO project and designer of its student portal.

●

Alexander Refsum Jensenius is a music researcher and research musician. His research focuses on why music makes us move, which he explores through empirical studies using different types of motion sensing technologies. He also uses the analytic knowledge and tools in the creation of new music, with both traditional and very untraditional instruments. Alexander is Professor of music technology and Deputy Director of RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion at the University of Oslo.

●

Sigurd Saue just left the position as associate professor in Music Technology at the Norwegian University of Science and Technology (NTNU). He has a background in acoustics, electrical engineering, music and theatre. Prior to his academic position, he worked partly with audio/seismic signal processing for the oil industry and partly as developer in electronic art projects in cooperation with composers and artists. His academic focus was interactive audio in various disguises: Sonification, real-time music performance, sound installations and game audio.