Facilitating Team-Based Programming Learning with Web Audio

Anna Xambó
Department of Music
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
anna.xambo@ntnu.no

Robin Støckert
Department of Mathematical Sciences
NTNU
Trondheim, Norway
robin.stockert@ntnu.no

Alexander Refsum Jensenius
RITMO
Department of Musicology
University of Oslo
Oslo, Norway
a.r.jensenius@imv.uio.no

Sigurd Saue
Department of Music
NTNU
Trondheim, Norway
sigurd.saue@ntnu.no

ABSTRACT

In this paper, we present a course of audio programming using web audio technologies addressed to an interdisciplinary group of master students who are mostly novices in programming. This course is held in two connected university campuses through a portal space and the students are expected to work in cross-campus teams. The course promotes both individual and group work and is based on ideas from science, technology, engineering, arts and mathematics (STEAM) education, team-based learning (TBL) and project-based learning. We show the outcomes of this course, discuss the students' feedback and reflect on the results. We found that it is important to provide individual vs. group work, to use the same code editor for consistent follow-up and to be able to share the screen to solve individual questions. Other aspects inherent to the master (e.g. intensity of the courses, coding in a research-oriented program) and to prior knowledge (e.g. web technologies) should be reconsidered. We conclude with a wider reflection on the challenges and potentials of using web audio as a programming environment for novices in TBL cross-campus courses and how to foster effective novices.

1. INTRODUCTION

The 21st century brings sophisticated information and communication technologies different from the 20th century, which is the reason why school curriculums are changing to a new culture of learning and the promotion of a new set of skills, the "21st century skills" [6, 11]. Programming can be considered an important component of technology literacy, which is one of the identified skills of the 21st century [11]. 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. In particular, Robins [15] distinguishes between effective and ineffective novices, and suggests to focus on promoting effective novices. A relevant factor is motivation, which can be enhanced with more interdisciplinary ways of learning, such as science, technology, engineering, arts, and mathematics (STEAM) education [7]. It has been reported that teaching programming based on STEAM promotes higher level of creativity [21] and can attract the interest of underrepresented populations in computing fields [8].

The NTNU-funded project Student Active Learning in a Two Campuses Organization (SALTO) aims to promote cross-campus learning as an open laboratory [17], where novel student-active learning strategies are investigated, such as computer-supported collaborative learning [16], flipping classroom [2] and team-based learning (TBL) [10]. Cross-site interaction is framed as a future scenario in education (e.g. teaching and learning environments) and work (e.g. working environments). The combination of simultaneous co-located vs. remote interactions is mediated through low-latency audiovisual and communication technologies.

A new international master has been launched within this educational scheme: Music, Communication, and Technology (MCT),[1] which is a master's program in collaboration between the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway, and the University of Oslo (UiO) in Oslo, Norway. The program centres around the field of music technology from a research perspective in a cross-campus setting. The students work in cross-campus teams, which are interdisciplinary, and they have different levels of expertise in STEM fields, such as programming.

As part of the master's program, most of the master courses are offered in a condensed format of two full-day weeks (eight days in total) so that students can focus on one subject at a time and can have more flexibility in their schedule. This can be especially useful for technical subjects, so that students focus on the same technology for a certain period of time. In our previous study on a physical computing workshop [20],

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

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

we found that the use of hybrid technologies to teach physical computing to mixed-levels cross-campus groups was a complicated environment for novices in programming. In this paper, we present instead the results of an intense course in audio programming, only taught with web audio technologies, which is a recommended tool for novices [1].

The research question that this paper addresses is: *What are the challenges and opportunities of using web audio to teach audio programming to a mixed cohort of students mostly novices in programming?* We collected students' feedback and reflected on the students' outcomes and comments. Overall, the results indicate that we can recommend to use web audio to teach audio programming to a mixed group of mostly novices, if there is a fair combination of individual and group work, the students and teacher use the same code editor for consistency, and it is possible at all times to share the code between the teacher, smaller groups and the whole class to promote debugging in a collaborative context. However, pre-knowledge should be required in web technologies and basic programming, as well as focus the teaching on strategies instead of knowledge, in order to promote "effective novices".

2. BACKGROUND

Our previous research highlighted how collaborative music live coding (CMLC), which refers to the combination of the music improvisation practice of live coding and collaborative programming, is a promising learning strategy of programming [19]. In a more recent study on teaching physical computing to mostly novices [20], we also found that programming is a difficult skill to learn in a short period of time, as opposed to prototyping. As noticed by Robins et al.: *"It is generally accepted that it takes about 10 years of experience to turn a novice into an expert programmer."* [15, p.137] and therefore Robins et al. recommend to present material that addresses basic program design.

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 [1] to teach creative coding concepts through music by using different web audio libraries and frameworks, such as Gibber [14] and Tone.js [9]. 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 (e.g., trigonometry and wave motion, additive synthesis, Fourier analysis, pitch spiral, noise spiral) to fine arts students [3]. EarSketch is a learning environment and curriculum that promotes to learn how to code by making music among middle school, high school and undergraduate students, an environment reported to broaden participation in computing and music, particularly women [8]. Codecircle is a browser-based system for learning creative coding that supports real-time graphics and sound [21].

In this paper, we present a course of audio programming using web audio technologies addressed to an interdisciplinary group of master students, who are mostly novices in programming. We used a similar approach to [1] of exposing the students to different libraries and frameworks, with the final aim of creating their own project. However, our course was longer so we also included native web audio to facilitate the understanding of these higher-level environments. We promoted both individual but also cross-campus group projects.

3. THE COURSE

3.1 Context

The MCT4048 Audio Programming course is an elective master course held in Spring that aims 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 are a number of reasons for choosing web audio technologies for this course, namely:

- It is written in JavaScript, one modern programming language.
- It is easy to sketch ideas and get prototypes built.
- It is easy to test, implement, and distribute.
- It showcases the fundamental concepts of audio programming.
- It gives room for artistic expression.
- It is an employable skill.
- We will be hosting the Web Audio Conference 2019.[2]

The students have a multidisciplinary background ranging from digital humanities, to music technology, to engineering. Their backgrounds in programming skills are varied, predominantly novices. In this course edition, the number of total students was 11 between the two campuses, with 4 students in the Oslo site and 7 students in the Trondheim site, and two women. The group was international with students from Europe and Asia.

3.2 Curriculum

This course combines lectures and hands-on activities, whose slides and code are available online.[3] The lectures provide an overview of the fundamental concepts of audio programming. The hands-on workshops are based on building web applications using web audio technologies, both individually and in team. The evaluation of the course consists of measuring the daily activity and two mini-projects that incorporate the theory and practice seen in class.

Given that this course belongs to a research-based master, it combines research-based activities with development activities. The course spans 7 hours per day during 8 days (56 hours in 2 weeks). We allocate some time at the beginning of each session to set up the computers with the tools for the tutorial of the day. The first week is named "The Fundamentals" and the students are asked to develop an individual mini-project. This allows each student to work individually, familiarize themselves with the programming environment and find their way of expression. The second week is named "The Extensions" and the students are asked to develop a group mini-project. Next we show the general outline of the course by day:

1. **Introduction Day**: Paper discussion about languages for computer music [5]. Discussion about what is and why to use the Web Audio API. Discussion and exercise about pseudocode.

[2]https://ntnu.edu/wac2019
[3]https://github.com/axambo/audio-programming-workshop

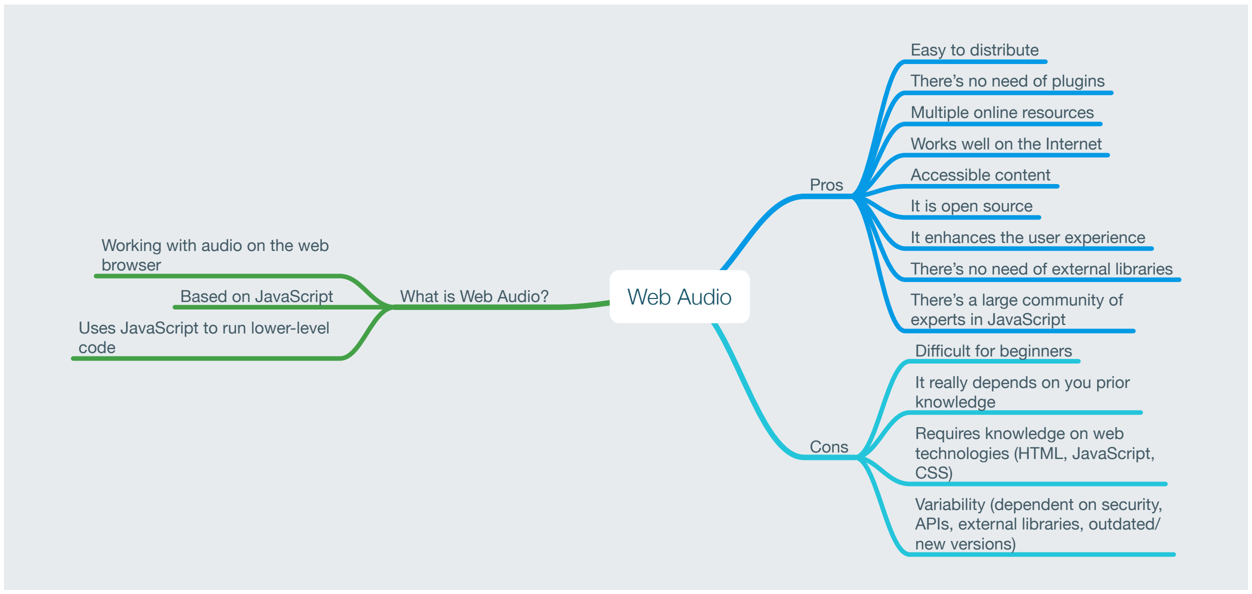

Web Audio
What is Web Audio?
Working with audio on the web browser
Based on JavaScript
Uses JavaScript to run lower-level code
Pros
Easy to distribute
There's no need of plugins
Multiple online resources
Works well on the Internet
Accessible content
It is open source
It enhances the user experience
There's no need of external libraries
There's a large community of experts in JavaScript
Cons
Difficult for beginners
It really depends on you prior knowledge
Requires knowledge on web technologies (HTML, JavaScript, CSS)
Variability (dependent on security, APIs, external libraries, outdated/ new versions)

Figure 1: Mind map generated in class from a group discussion about what is web audio.

2. **Day 1**: Warm-up activity: what is Web Audio, pros and cons (see Figure 1). Tutorial of playing sounds and individual mini-project development.
3. **Day 2**: Warm-up activity: a round of one sentence each about something learned the previous day in class. Tutorial of dealing with time using Tone.js [9], and individual mini-project development. Speedy presentations of the mini-projects.
4. **Day 3**: WAC paper presentations (first half of the group, one paper per student). Tutorial of dealing with sound effects and individual mini-project development. Speedy presentations of the mini-projects.
5. **Day 4**: WAC paper presentations part 2 (second half of the group, one paper per student). Tutorial of graphical user interfaces using NexusUI.js [18], and mini-project development. Final presentations of the individual mini-projects.
6. **Day 5**: Warm-up activity: Recap quiz of week 1. Tutorial of dealing with interactivity using Web MIDI API[4] and group mini-project development.
7. **Day 6**: Warm-up activity: what do we know about live coding. Tutorial of live coding using Gibber [14], and group mini-project development. Speedy presentations of the mini-projects.
8. **Day 7**: Tutorial of mobile music and responsive design, and group mini-project development. Speedy presentations of the mini-projects.
9. **Day 8**: Tutorial of AudioWorklets [4, 12, 13], and group mini-project development. Final presentations of the group mini-projects.

3.3 Outcomes

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 can be found online in the student-led MCT blog.[5]

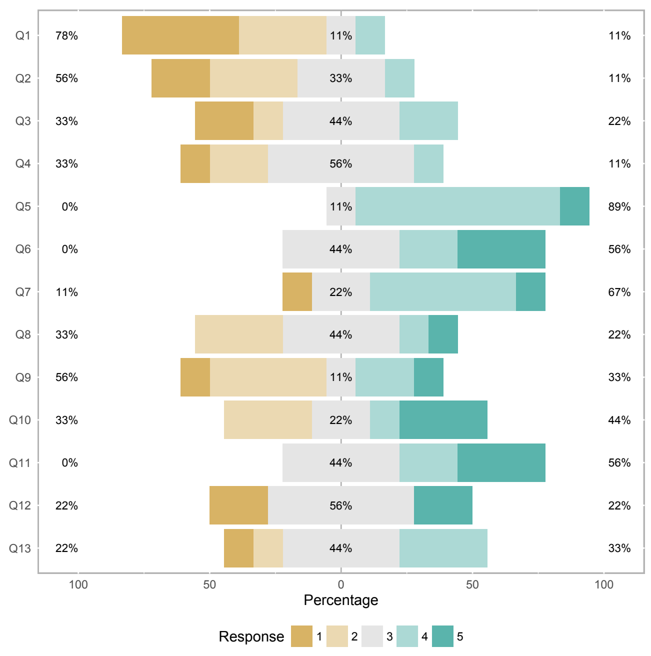
Next, a brief description of the three group mini-projects is provided:

- *Touch the Alien* (Team A) – A web audio synth which offers touchscreen functionality; a combination of oscillators, FM oscillator, delay, phaser, chorus, and filter on dry/wet slider; and an interactive canvas user interface to control the filters. The technologies used include Javascript with the Web Audio API, CSS, HTML5, the audio effects library Tuna, and a smartphone or tablet.
- *The Magic Piano* (Team B) – A piano that plays the right notes of the chosen melody regardless of whether the player hits the right or wrong key. The technologies used include Web MIDI API, NexusUI.js, Tone.js, JSON, CSS and a MIDI keyboard.
- *Convolverizer* (Team C) – Real-time processing of ambient sound, voice or live instruments. The technologies used include P5.js, a sound card or audio interface, a guitar, and a Shure SM57 microphone.

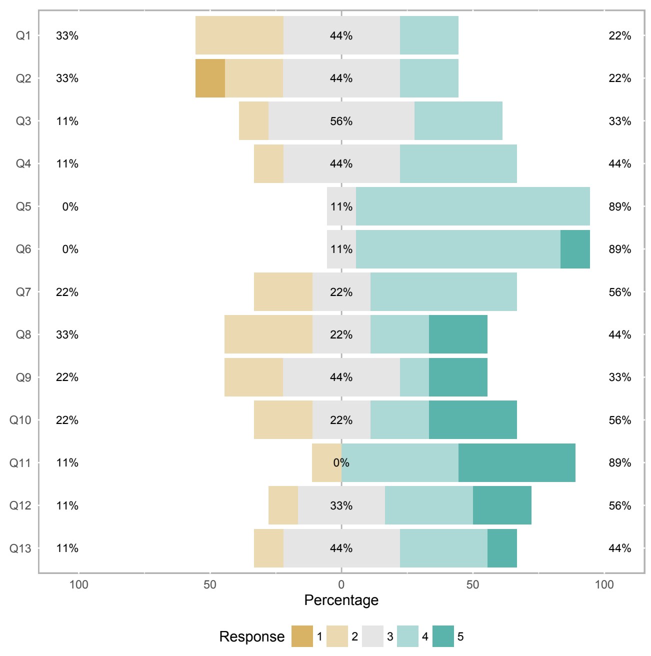
The three group mini-projects were successful in completion. Although varied in theme, there are some similarities: They (1) build on the code developed individually during the previous week (except for Team C who built on knowledge from the physical computing workshop), with the challenge of merging their code and working with collaborative coding approaches, both co-located and remote; (2) use a range of web audio and web technologies, some of them seen in class but some of them explored autonomously or in previous courses, which showcases the understanding of the spirit of prototyping in finding the best tools and combining them to convey a project idea; and (3) combine software with hardware, building on previous knowledge from the physical computing workshop.

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

[5]https://mct-master.github.io/audio-programming



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



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

Figure 2: 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.

4. STUDENTS' FEEDBACK

Adapted from our previous physical computing workshop [20], we distributed among students a voluntary pre-questionnaire and post-questionnaire with the same 5-point Likert-item questions. 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 that asked the level of agreement (1 = strongly disagree; 2 = disagree; 3 = neutral; 4 = agree; 5 = strongly agree) about 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). They were also asked their level of agreement of the extent to which they can understand the purpose of audio programming (Q11), describe the process of programming an interactive musical prototype (Q12), and apply the technique of programming an interactive musical prototype to their work (Q13). The questionnaire also included three open questions about what the students liked best and least about the course and how the course could be improved.

We obtained responses from 11 students in total, out of which 9 students responded the paired questionnaire ($n = 9$). Of the course, the students 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).
- **Mixed groups and research-based 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).

Students' suggestions on how to improve the course included:

- **Avoid intensive course format** (4 occurrences): *"making it less intense and more spread during the term"* (U8); *"extend to more then 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 needed to evaluate"* (U3); *"the way it was this year would suit more for students with prior understanding of programming to some level"* (U9); *"more clearly defined incremental tasks related to the curriculum, which should also have the possibility of extra challenges for those that are on a higher level"* (U2).

Figure 2 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 [20], which indicate that the pedagogical techniques are in a positive direction. The level of confidence of programming (Q1), together with the level of prototyping (Q3) and instrument building (Q4) slightly improved, providing at least little more confidence. This contrasts with the level of confidence of programming achieved in the physical computing workshop, which was one of the less developed skills. The intention to continue STEM courses (Q8) and STEM education (Q9) improves slightly, again in alignment with our previous study. 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 master's courses. The level of confidence of individual work (Q7) changed from extreme to moderate positive and negative opinions. Individual work was an important component of this course during the first week, 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. REFLECTIONS

This section reflects on the students' feedback presented in the previous section combined with the reflections of the teacher of the course (first author). Overall, learning how to program is a slow endeavor, and team-based programming can be a helpful approach. There are five prominent themes that are worth of discussion: (1) individual vs. group work; (2) shared coding experiences; (3) web audio vs. web technologies; (4) research vs. code; and (5) fast-paced vs. slow-paced teaching.

5.1 Individual vs. Group Work

The sequence of individual work during the first week and group work during the second week generally worked well. The students developed an individual language and style using web audio technologies, which was helpful for the collaborative work in the following week. This is similar to the need of individual rehearsal with your own musical instrument to first define your voice before meeting with a group to collaborate musically. Prior knowledge to the course that could be acquired includes basic programming and how to merge code from different contributors. The additional asset of this course, compared to other similar courses (e.g., [1]), is using TBL to solve problems that have a higher level of complexity than when working individually. The shareable nature of web technologies (e.g. easy exchange and execution of code snippets) seems to align well with TBL activities.

5.2 Shared Coding Experiences

Although not directly reported in the students' feedback, 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. Considering that this course should be taught by only one teacher to cross-campus scenarios of at least one co-located group and one remote group, the real-time audiovisual and screen share communication becomes crucial. With this approach, there were occurrences of expert students helping novices, therefore a student's problem could often become a group problem, and everyone was learning from each other, in terms of both errors and discoveries. Both the infrastructure of the portal with network-connected working rooms and the use of the same programming environment and language with shareability capabilities seemed to help.

5.3 Web Audio vs. Web Technologies

Novices were sometimes overwhelmed with the need to learn not only web audio, but also web technologies. By contrast, experts were sometimes expecting more focus on web audio and put less attention to web technologies. Although web audio can be a suitable tool for learning audio programming, novices should fulfil the pre-knowledge requirements of already knowing the basics about web technologies. Similarly, the basics of programming should be also a pre-requisite in order to balance better novices vs. experts. As suggested by Robins [15], it is important to focus on teaching strategies as opposed to knowledge in order to promote "effective novices", who become proficient in solving programming problems.

5.4 Research vs. Code

Teaching an audio programming course at a research-based master implies combining hands-on practical work with theory and reflection. However, we found that novices were expecting to also learn how to code, whilst experts were expecting to code more. The meaning of a research-based course in audio programming should probably be emphasized at a master level and reinforced at a course level, so that students' expectations are closer to the nature of the master.

5.5 Fast-paced vs. Slow-paced Teaching

The students' feedback suggests to expand or reconsider the intensity of the course. An option can be to link better this course with other courses, so that the students perceive a continuous learning path across courses. Given that some students report the need of more formal teaching or more time to consolidate their knowledge, perhaps there could be curated resources that help the students to continue by their own if they want to, similar to the curriculum of EarSketch [8].

6. CONCLUSIONS

In this paper, we presented an audio programming course using web audio technologies targeted to an interdisciplinary group of master students who are mostly novices in programming. Students' feedback and teacher's reflections indicated that web audio technologies is a suitable approach to novices in programming if web technologies and basic programming are requested as prior knowledge. The projects resulted from working in teams have shown the potential of addressing complexity with creativity and collaboration, which was partly possible due to the prior 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 issue. We expect that in the second edition of this course we will foster better "effective novices", so that the two intense weeks can be even more fruitful. In future work, we plan to study with a larger number of students; understand which features of web audio enhance collaboration in TBL activities; and investigate whether web audio works better for TBL activities compared to native environments (e.g. C++ audio library, Max or SuperCollider). We also hope to explore more the potential of teaching audio programming with web audio for STEAM education, TBL and distance learning.

7. ACKNOWLEDGMENTS

The authors wish to thank the students who participated in the course. Also thanks to the MCT teachers Daniel Formo, Anders Tveit and Kristian Nymoen for their technical help. This work was partially supported by the NTNU SALTO project (80340480).

8. REFERENCES

[1] J. Allison, D. Holmes, Z. Berkowitz, A. Pfalz, W. Conlin, N. Hwang, and B. Taylor. Programming Music Camp: Using Web Audio to Teach Creative Coding. In *Proc. Web Audio Conference*, 2016.

[2] J. Bergmann and A. Sams. *Flip Your Classroom: Reach Every Student in Every Class Every Day.* International Society for Technology in Education, Eugene, OR, USA, 2012.

[3] I. G. Burleigh and T. Schaller. Quint.js: A JavaScript Library for Teaching Music Technology to Fine Arts Students. In *Proc. Web Audio Conference*, 2015.

[4] H. Choi. AudioWorklet: The Future of Web Audio. In *Proc. International Computer Music Conference*, 2018.

[5] R. B. Dannenberg. Languages for Computer Music. *Frontiers in Digital Humanities*, 5:26, 2018.

[6] C. Dede. Comparing Frameworks for 21st Century Skills. *21st Century Skills: Rethinking How Students Learn*, 20:51–76, 2010.

[7] J. Maeda. STEM + Art = STEAM. *The STEAM Journal*, 1(1):34, 2013.

[8] B. Magerko, J. Freeman, T. Mcklin, M. Reilly, E. Livingston, S. Mccoid, and A. Crews-Brown. Earsketch: A STEAM-based Approach for Underrepresented Populations in High School Computer Science Education. *ACM Transactions on Computing Education*, 16(4):14, 2016.

[9] Y. Mann. Interactive Music with Tone.js. In *Proc. Web Audio Conference*, 2015.

[10] L. K. Michaelsen, A. B. Knight, and L. D. Fink. *Team-based Learning: A Transformative Use of Small Groups in College Teaching.* Stylus Pub, 2004.

[11] B. Pearlman. Making 21st Century Schools: Creating Learner-Centered Schoolplaces/Workplaces for a New Culture of Students at Work. *Educational Technology*, pages 14–19, 2009.

[12] C. Roberts. Strategies for Per-Sample Processing of Audio Graphs in the Browser. In *Proc. Web Audio Conference*, 2017.

[13] C. Roberts. Metaprogramming Strategies for AudioWorklets. In *Proc. Web Audio Conference*, 2018.

[14] C. Roberts and J. Kuchera-Morin. Gibber: Live Coding Audio in the Browser. In *Proc. International Computer Music Conference*, 2012.

[15] A. Robins, J. Rountree, and N. Rountree. Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, 13(2):137–172, 2003.

[16] G. Stahl, T. D. Koschmann, and D. D. Suthers. *Computer-Supported Collaborative Learning.* na, 2006.

[17] R. Støckert, A. R. Jensenius, and 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. In *Proc. International Conference of Education, Research and Innovation*, pages 5831–5840, 2017.

[18] B. Taylor, J. T. Allison, W. Conlin, Y. Oh, and D. Holmes. Simplified Expressive Mobile Development with NexusUI, NexusUp, and NexusDrop. In *Proc. New Interfaces for Musical Expression*, pages 257–262, 2014.

[19] A. Xambó, G. Roma, P. Shah, T. Tsuchiya, J. Freeman, and B. Magerko. Turn-taking and Online Chatting in Co-located and Remote Collaborative Music Live Coding. *Journal of the Audio Engineering Society*, 66(4):253–266, 2018.

[20] A. Xambó, S. Saue, A. R. Jensenius, R. Støckert, and Ø. Brandtsegg. NIME Prototyping in Teams: A Participatory Approach to Teaching Physical Computing. In *Proc. New Interfaces for Musical Expression*, 2019.

[21] M. Yee-King, M. Grierson, and M. d'Inverno. STEAM WORKS: Student Coders Experiment More and Experimenters Gain Higher Grades. In *Proc. IEEE Global Engineering Education Conference*, pages 359–366. IEEE, 2017.