BEATBLOX: A VISUAL BLOCK-BASED APPROACH TO INTEGRATING MUSIC AND COMPUTER SCIENCE EDUCATION

T. Ebiwonjumi, W. Hedgecock, D. Jean, G. Barnard, S. Kittani, B. Broll, A. Ledeczi

Vanderbilt University (UNITED STATES)

Abstract

As computing increasingly permeates society, a solid Computer Science (CS) background emerges as a vital skill for all students. However, traditional CS instruction often struggles to engage diverse learners. To address this challenge, we introduce BeatBlox, an innovative music-based framework built upon the NetsBlox platform. BeatBlox integrates CS concepts with music, offering a captivating and accessible approach to learning. Through co-design sessions with students and educators, BeatBlox was tailored to reflect diverse cultural interests and musical preferences. Its features allow students to create music, visualizers, and sound projects, and even leverage augmented reality for instrument-free composition. BeatBlox fosters collaboration through its built-in networking support enabling both collaborative coding across multiple computers and immersive class projects such as digital orchestras playing synchronized music. By fusing CS education with music, BeatBlox provides a compelling pathway to engage students of all backgrounds and musical abilities, promoting inclusivity in the learning process.

Keywords: Computer science education, music, block-based programming, distributed computing, interdisciplinary engagement

1 INTRODUCTION

Computing is fundamentally transforming every aspect of our lives, and there is a growing consensus that Computer Science (CS) is a 21st-century skill that all students should be exposed to. Nonetheless, CS is often considered dry, difficult, and only appealing to a small segment of the population, typically white and asian male students, partly because traditional CS curricula tend not to include content that historically underrepresented groups find interesting, motivating, or relevant. For example, studies have shown that women typically find CS topics more meaningful when connected to cultural issues and integrated with other disciplines [1]. Existing content also fails to account for the individual learning variations of the students themselves. On top of that, CS educational mandates are being implemented at an accelerating pace at the same time as budget and staffing cuts, leading to difficulties in finding qualified CS teachers when their backgrounds and expertise are often in other subject areas. It is therefore important to present CS from a variety of perspectives [2] and to make connections to other subjects [1]. We found the most successful classroom projects are ones in which students are able to connect to concepts relevant to their own communities, explore subjects that are personally motivating and meaningful to them, and take ownership over the design process. This led to our inclusion of topics like climate change, movies, public health, and social justice in a previous work [3], which we found to greatly increase student interest and engagement in computing.

Specifically, this previous work involved the development of a tool called NetsBlox, a browser-based visual programming environment, and corresponding cloud infrastructure, which was designed to present a select set of distributed coding primitives as intuitive and easy-to-grasp visual abstractions, capturing the essence of important concepts while hiding unnecessary complexity [4, 5]. This tool provides access to many online data services including Google Maps, the Online Movie Database, USGS earthquake data, NOAA climate change data, and many others. Built-in message passing allows students to create real distributed programs like online multiplayer games, chat rooms, or other social applications. We found that these abstractions and ties to contemporary topics are the key to making CS more engaging and relevant to students' lives.

In this work, we focus on perhaps the most broadly appealing topic of all: music. We recently developed an extension to NetsBlox called BeatBlox, a music-based framework for teaching computer science. By connecting CS concepts to a subject already of interest to people across diverse socioeconomic backgrounds, the learning process becomes more fun, engaging, and accessible to both educator and student.

2 BACKGROUND

When teaching computer science to K-12 students, many encounter a significant level of fear and aversion toward programming projects. This apprehension is often attributed to the complex nature of traditional programming languages such as C++, Python, and Java. Visual programming [4] offers an appealing alternative for imparting computer science concepts, particularly through block-based programming. Projects like Scratch [6] have demonstrated the efficacy of block-based programming, albeit with limited capabilities compared to general-purpose programming languages. Snap*!* [7] preserves the accessibility of Scratch while enhancing its functionality to resemble that of general-purpose programming languages. It achieves this by incorporating more advanced features such as support for recursion, creating custom blocks (functions), more advanced data types, and more.

Despite these advancements, Snap*!* lacks a crucial component: simple abstractions to access web-based resources and facilitate the creation of networked projects. To address these limitations, NetsBlox [4, 5] was developed, introducing two primary features: Remote Procedure Calls (RPCs) and message passing. RPCs enable users to retrieve internet-based data, while message passing allows users to send and receive data between NetsBlox projects over the internet.

With these additional features, NetsBlox offers a comprehensive platform for teaching a wide range of topics, from introductory courses to more advanced ones. Students have been able to create Weather Apps using the Google Maps and Weather services, as shown in Figure 1, multiplayer games using message passing, and even visualize scientific data such as COVID-19 case counts.

Figure 1. Simple NetsBlox project showing a map of our location as the background and displaying current weather conditions. Top script: stage, bottom script: sprite.

While NetsBlox has made significant strides in offering distributed and embedded computing through a block-based programming format, it still lacks an exciting feature that could engage a diverse group of students: music. Projects like SonicPi[8] and EarSketch[9] utilize traditional programming languages such as JavaScript and Python to teach computing concepts through music. SonicPi operates as a downloadable desktop program, running in a specialized environment that allows users to write a few lines of code and generate complex sounds. On the other hand, EarSketch enables users to write code in JavaScript or Python and produce sounds directly on the web.

BeatBlox aims to leverage a block-based programming language within a web environment (NetsBlox) to teach computer science through music. By integrating music composition and coding in a user-friendly manner, BeatBlox aims to bridge the gap between creativity and technology for students of all backgrounds. BeatBlox was constructed with the Web Audio API [10] as its foundation, which is the contemporary standard for controlling audio on the web and is supported by most modern browsers. By leveraging the Web Audio API, BeatBlox can swiftly and efficiently synthesize sounds and manage audio playback.

3 BEATBLOX

NetsBlox inherits its sound functionality from Snap*!*, allowing users to play sounds and individual notes from a selection of oscillator-based sound sources: sine, square, sawtooth, and triangle. Additionally, Snap*!* features a native sound object, enabling students to upload sounds and play them in their projects. Snap*!* sound blocks were developed before the introduction of the Web Audio API, resulting in limited

functionality and sound quality. To achieve our goal of teaching computer science through music, we recognized the need to enhance the audio capabilities of the platform by building our own implementation of music blocks.

To ensure clarity in terminology, let's establish definitions for key terms used in BeatBlox. In this context, a "sound" refers to any pre-recorded audio loaded into BeatBlox. A "note," akin to a musical note, is played by a virtual instrument within BeatBlox. A "virtual instrument" constitutes a playable source that interprets note names and generates sound, e.g., Grand Piano, Electric Guitar, etc. Lastly, a "sound effect" serves to augment or modify the sonic characteristics of the produced sound.

BeatBlox extends NetsBlox with two additional features: 1) programming blocks for generating music and sound effects, and 2) a catalog of hundreds of curated pre-recorded sounds. Using the new music-based blocks, students can carry out musical tasks such as selecting an instrument to use for playback, programming individual notes and note durations, adding sound effects, adjusting tempo, or even interacting with external musical devices such as digital keyboards. Students can even record completely new audio into a variable and manipulate it using programming primitives in the exact way used in more traditional logic-based programs. For example, a student can select the “Grand Piano” instrument and programmatically implement “Für Elise” or any of their favorite piano pieces.

The purple blocks on the left-hand side of Figure 2 represent a selection of the new musical functionality built into BeatBlox. Some examples of these blocks and their corresponding functionality include:

- Set instrument: selects an instrument from the list provided.
- Play note: plays a specific note, using the selected instrument.
- Play sound: plays the sound provided to the block.
- Track effects: applies a sound effect.
- Set tempo: sets the tempo of the project.

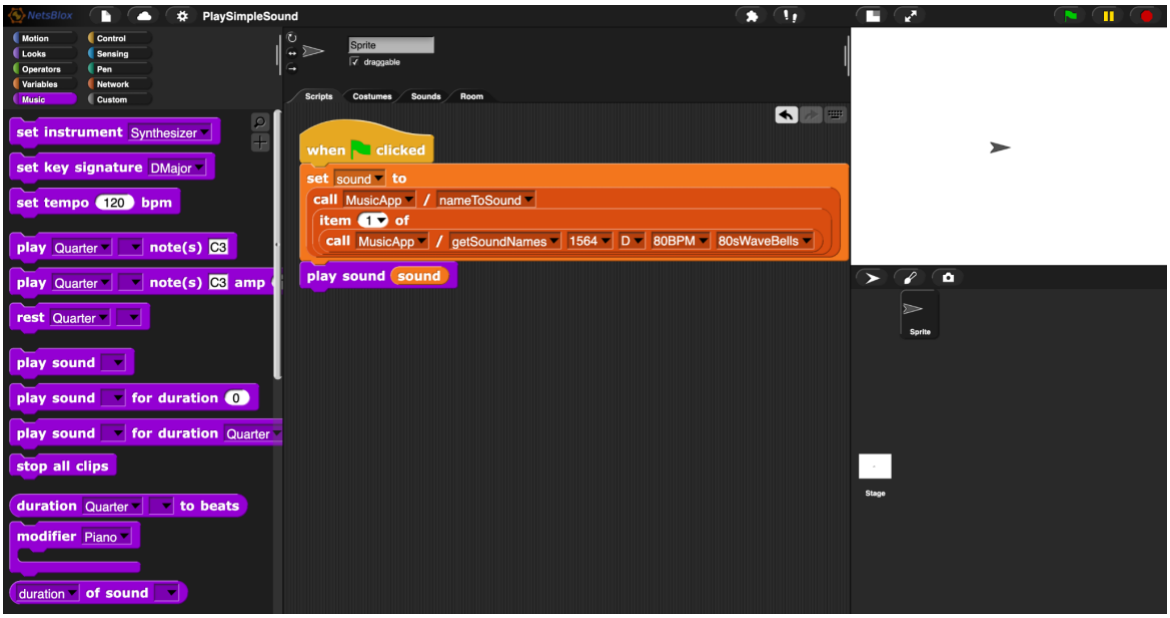

Figure 2. Examples of using BeatBlox RPCs and play block.

Students have access to a large library of pre-recorded sounds to incorporate into their projects through a set of RPCs that allow them to search based on key characteristics of the sound and then to retrieve them from the server. The aim is to empower students to dive into music creation while bypassing the need for extensive knowledge of music theory. Figure 2 illustrates an example project retrieving an audio clip from the sound library, storing it into a variable, and playing the sound.

One of the major features of BeatBlox is its implementation of an internal clock designed to synchronize music blocks seamlessly. Using the original Snap*!* sound blocks, an issue arose when attempting to play multiple blocks concurrently, resulting in the audio becoming unsynchronized. To address this challenge, our internal clock intervenes, aligning music blocks and ensuring audio synchronization. Our internal clock interacts with the Web Audio API to retrieve the current time, subsequently generating an

internal variable. This time data is then distributed across all the blocks within the project. When a block intends to play audio, it patiently awaits the completion of any preceding blocks, scheduling its playback to commence afterward. This ensures that blocks consistently maintain synchronization with one another, thereby preserving the integrity and cohesion of the musical composition.

Another timing issue exists when attempting to play synchronized audio on multiple computers, for example, to create a digital orchestra. The clocks on separate computers are not in perfect alignment with one another or with the actual current time. If we want to start two virtual instruments at the same time on two different computers, we need to have an underlying mechanism to ensure synchronicity. BeatBlox implements a time synchronization algorithm that aligns each client's time to that of the server with better than 10 msec accuracy. In turn, a project playing one instrument can send a message to all other projects playing different instruments at the exact time when everybody needs to initiate playback. This ensures that they all start playing simultaneously, at least to the human ear. The same idea can be used to implement distributed sound effects across multiple computers.

Just like most other block-based programming environments, NetsBlox has graphical objects called Sprites. Sprites can be programmed to perform specific actions and behaviors within a project. When creating games, students typically create a Sprite for every element or character within their game and can give each Sprite code to control its actions. Sprites also have their own visual representation that can be changed individually irrespective of one another. An important design choice made in BeatBlox was to make each Sprite represent its own Instrument/Track. With each Sprite acting as one instrument within the context of an orchestra, it allows users the ability to have a visual representation of each instrument and its own scripts, each playing its own sound. This design choice enables the application of sound effects at the Sprite level. When an effect block is invoked, it solely affects the sounds emitted from the Sprite in which the block is called. As a result, multiple effects can be layered within the same BeatBlox project without interference allowing for enhanced creativity and sonic experimentation.

4 METHODOLOGY

To ensure the development of the right features and to minimize creator bias, we conducted a series of focus groups where we collaborated with secondary students and educators to discover their prevailing pain points. These co-design sessions underscored the need to provide a variety of materials and functionality to ensure BeatBlox's accessibility to a diverse group of students.

One such session took place during the Fall 2023 semester, involving 20 undergraduate students. The students had some familiarity with NetsBlox already, so we were able to focus on introducing them to BeatBlox directly. Most participants were freshman students who closely aligned with one of our target groups of high school students, so we considered this a valuable case study. During the focus group, students were introduced to BeatBlox and given three tutorial tasks to complete, each building upon the previous one.

To begin, the students were granted approximately half an hour to interact with and explore BeatBlox and to work through the three tutorials. Upon completion, the students were divided into four groups based on their musical background, ranging from None (no musical background) to High (extensive musical theory background). Within their groups, participants were asked to provide feedback on their experiences with BeatBlox, using green notes to highlight positive aspects and blue notes to articulate constructive improvement suggestions. An example of some of these notes is provided in Figure 4. By grouping participants based on their musical background, we aimed to boost their confidence in providing feedback and ensure the representation of a wider user group.

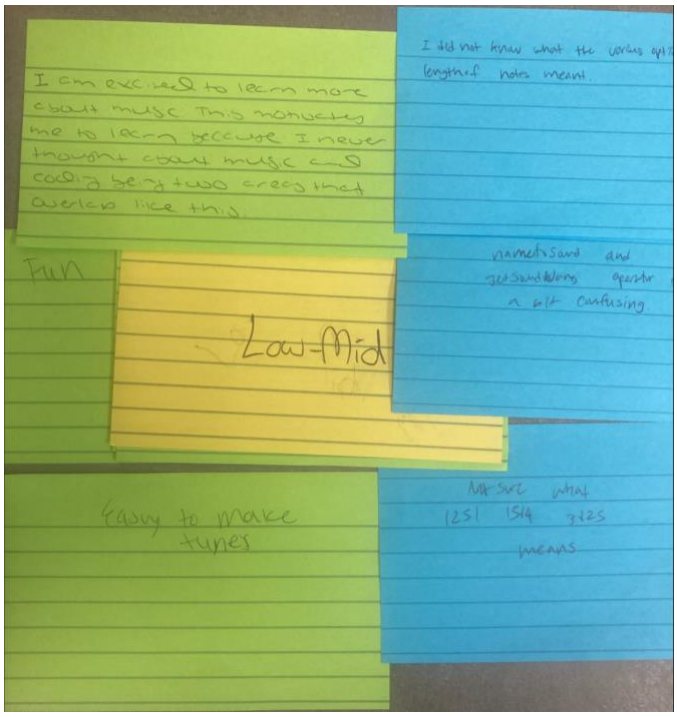

Figure 4. Feedback notes from the Low-Mid group in the co-design session

Following the feedback collection, we conducted a persona creation exercise. Each group was tasked with creating a persona—an imaginary middle-school-aged student embodying the common characteristics of their group as determined by exploring the commonalities of what they wrote on their notecards. Subsequently, they were asked to devise a curriculum to teach their persona if-statements and loops using BeatBlox. This exercise aimed to encourage participants to think more deeply about how they might connect the musical features and framework to basic programming concepts and to express candid feedback.

After creating their personas and curricula, we came back together to discuss each group's findings, along with overarching feedback from the session. In addition to helping us amass a vast amount of knowledge and insights into the various ways that learners tend to interact with curricular tools and materials, the insights gathered from this focus group informed the creation and refinement of various features within BeatBlox, which were then tested and validated in a subsequent session with middle school teachers.

During this follow-up session, we went through much the same process as before; however, this session revolved around gaining insights from actual educators about how they might use the tool as part of their classroom curriculum, and more importantly, how it might address the interest-based, motivational, and behavioral issues they face on a daily basis in their classrooms. Some of the key takeaways from this session are shown in Figure 5.

An interesting piece of feedback that kept recurring during this session was a concern that many students in underprivileged schools typically lack home access to the same types of devices or instruments available at school. As a means of directly addressing this challenge, we came up with the idea to introduce "augmented reality" instruments such as a theremin, air guitar, or paper drums into the framework. This feature enables students to stay engaged with their projects at home and on their own time with nothing more than their laptop (which is typically provided by the school to every student) and a piece of paper.

Both students and educators alike expressed that allowing students to experiment with different musical genres would be crucial to making our tool accessible to students from diverse backgrounds and cultures. One major feedback received from all groups was the necessity of ensuring that BeatBlox is usable for students with limited musical backgrounds. This led to the simplification of many musical blocks and the addition of more intuitive ways to input note names and durations.

Students also expressed a desire for different ways of interacting with music in terms of how involved they want to be in the music creation process itself. Namely, they wanted the ability to manipulate music

that already exists or to utilize music as the basis for other types of projects. To that end, we incorporated ways for students to generate cross-disciplinary projects, as will be shown in the following section.

Status	Implemented Feature or Concept
Does Well	Engages creativity and creates a tangible output
	Addresses interest and motivation by staying highly relevant to teenagers' lives
	Increases confidence by being very forgiving in terms of both functionality and producing good-sounding results
Needs Work	Too many options create feelings of confusion and being overwhelmed
	Overly verbose instructions are problematic for keeping attention and for addressing learner diversity regarding language skills
To Incorporate	Cross-disciplinary learning projects
	Exemplars: teenagers won't do anything until they know what the end product is

Figure 5. Design notes from a co-design session with Middle School teachers

Finally, an anonymized exit survey was given to participants across all sessions to capture their final thoughts, concerns, and feedback while still fresh in their minds. A small overview of the types of information collected through these surveys is shown in Figure 6:

Gender	Race	Coding Exp.	Coding Lang.	Instrument	Years Played	What aspects of the tutorials did you find intimidating?	What aspects of the tutorials helped you most to understand a related CS concept?	What aspects of the tutorials helped you most to understand the musical concepts?	How do you prefer to learn?	How do you prefer to learn?	Do you think you would have enjoyed an introduction to CS in middle school using BeatBlox?	What features, improvements etc. would you like to see added?	If we had more time, what else would you have liked to try to do with BeatBlox?	If your class were to use BeatBlox to learn CS concepts, what are the top two musical genres that you would be most excited to code with?	When usin learning, w prefer to in music that
Female	American Indian or Alaska Native	None	None	None	0	It was very wordy.	The budling aspect.	Showing the types of instruments and cords you could choose.	Through guided tutorials	In-person with a group or team	2	N/A	Further details on creating more projects	Pop and Rap	A mixture
Female	White	Some	None	none	0	I found the incorporation of musical terminology intimidating because I have minimal knowledge of stuff like notes and chords.	I think the in-depth instructions and the step-by-step processes helped a lot. For example, the broadcasting walkthrough.	Um not really any but I am tone deaf so music is hard and discouraging	little bit of both	In-person with a teacher or tutor	4	I think maybe like microphone stuff could be cool if that's not a thing	Make a song	Classical and rap	A mixture
Female	Asian	Some	Block-Based	none	2	saving variables and clips and incorporating the list aspect.	using the forever loop and repeat loop to play notes	the major and minor scales and the various effect options	Through guided tutorials	On your own with videos, books, and tutorials	4	drums and beat/metronome, more visual representation of each instruments	Mimic a known tune using the program	classical, jazz	A mixture
Male	White	Some	Java	n/a	0	calling the sounds in the different preloaded audio files	I liked the step by step walkthrough	I liked the basic explanations of notes and chords	Through experimentation	In-person with a teacher or tutor	4	Making the interface somewhat more easy to call different notes.	recording an audiofile	pop and classical music	A mixture
								The explanation of		On your own					

Figure 6. Overview of exit survey questions and results

5 RESULTS

In this section, we delve into various applications and capabilities of BeatBlox, each showcasing its ease of use and potential in an educational setting. This is organized according to the difficulty level of each BeatBlox project, illustrating how a potential curriculum could be structured.

To ensure BeatBlox can be utilized in an introductory computer science environment, our initial objective was to establish a low barrier to entry. Students should be capable of creating complex sounds using as few blocks as possible. As depicted in Figure 7, with a handful of blocks, we can produce an ethereal, satellite-like effect. This effect is achieved by combining three octaves of a C major scale, storing them into a variable, and subsequently playing back each note from this list in random order at varying volumes between 1 and 100 percent. Within this handful of blocks, we demonstrate: selecting instruments, adding audio effects to the track, and playing specific notes on the selected instrument.

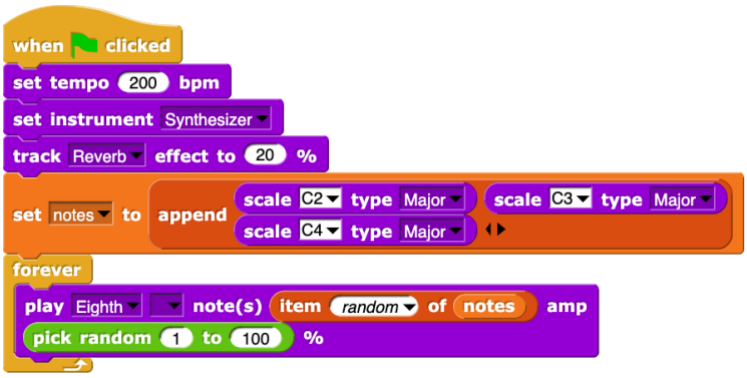

Figure 7. A BeatBlox script that creates an ethereal, satellite-like effect.

BeatBlox also has the capability of playing different segments of a musical composition as individual components of a whole. This is achieved by utilizing Sprites as described previously, each with separate scripts running concurrently. The project depicted in Figure 8 uses four Sprites to simultaneously play back all parts of a four-instrument composition, as if from a piano, saxophone, guitar, and bass.

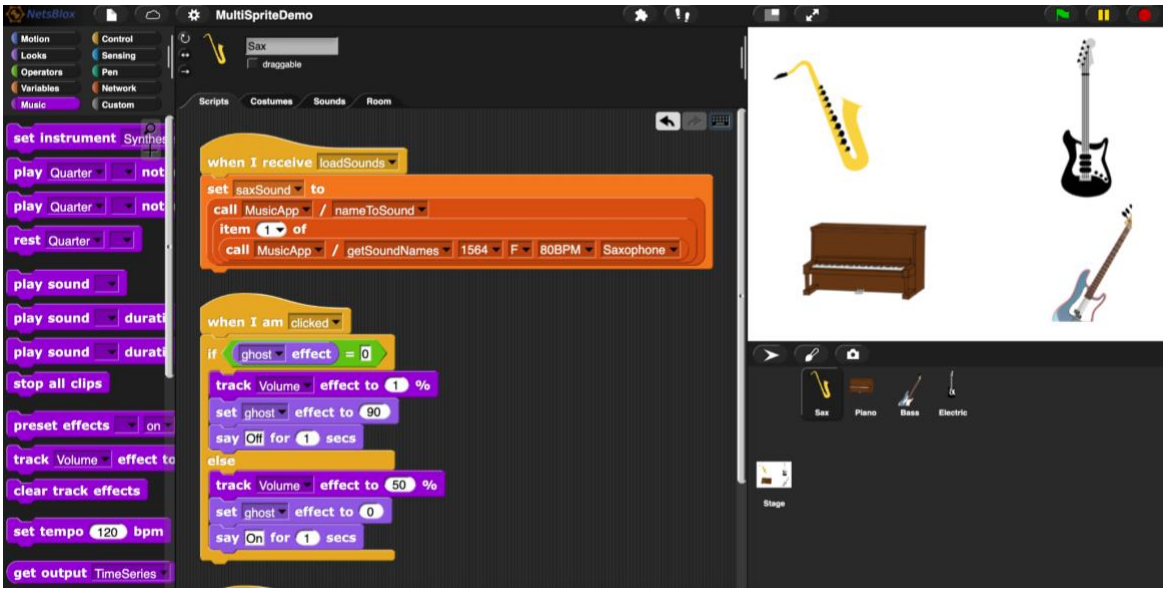

Figure 8. A BeatBlox project that plays four different instruments together

Within BeatBlox, we can also craft compelling and captivating visual representations of sounds. This fusion presents an ideal blend for teaching both musical and mathematical concepts. BeatBlox simplifies the process of utilizing a graphical stage area to generate engaging visualizations based on either the current power, audio, or spectral data being generated by sounds in real time. Figure 9 captures a snapshot of one way in which a student might tap into their artistic side to implement a graphical visualizer using raw time series data from an audio signal.

Similar to most modern digital audio workstations (DAWs), BeatBlox offers support for external audio and MIDI devices. In a DAW, commonly used for modern music production, users can connect external microphones and instruments to record audio. Additionally, with MIDI devices, they can record and playback a musician's performances. BeatBlox provides all of these functionalities within its web-based environment. Students with a more advanced musical background will find this feature both captivating and useful. Figure 10 displays the blocks used to select input devices, capture performances, and playback recorded audio.

Another accessibility-based function allows students to access sensors and run programs on their smartphones, allowing them to do things like create a maraca instrument. For schools that provide students with physical instruments, like keyboards, we integrated the capability to access external devices into BeatBlox, allowing students to generate and record real instrumental music on top of their programmed compositions as shown in Figure 11.

Figure 9. A BeatBlox project showing circular visualization based on time series data

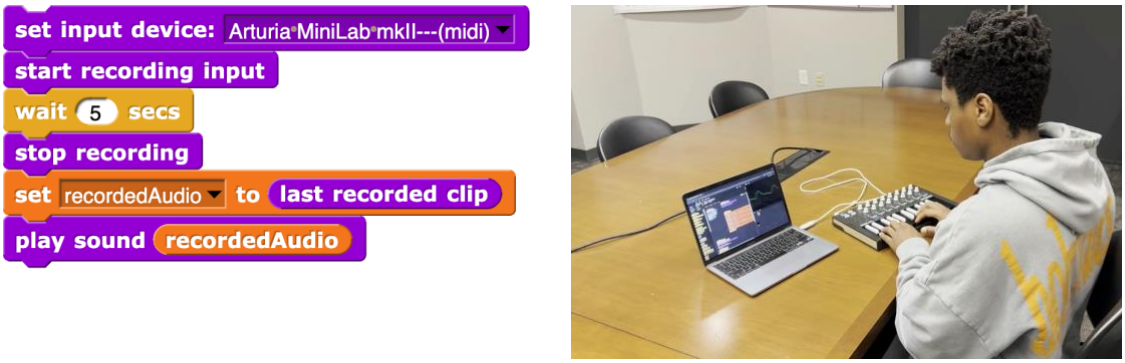

Figure 10. A BeatBlox user capturing their performance on a MIDI device

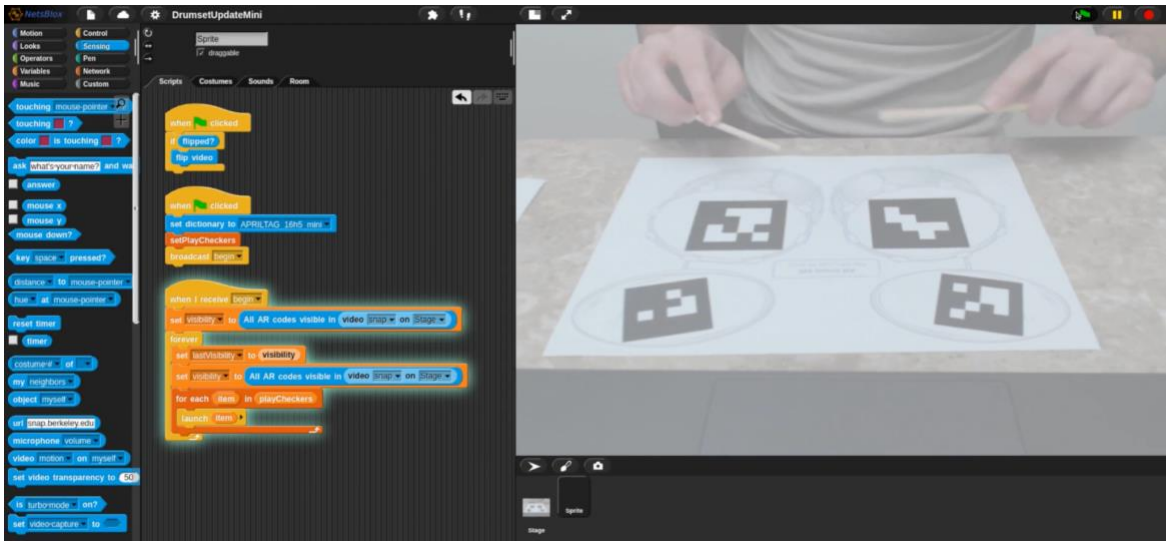

Figure 11. BeatBlox project with Augmented Reality drums

6 CONCLUSIONS

The ultimate metric for success for BeatBlox is whether its adoption causes students who would normally shy away from computer science to feel empowered to pursue studies in this field. While our initial focus was meant to be middle school students, after developing our prototype, we found that it had a broader reach than initially expected. We like to say that it has a low floor and a high ceiling. That is, it has capabilities to allow a novice 5th grader to learn CS concepts through the lens of music, but also the ability to create projects that would be challenging even for an experienced high schooler.

One of the key activities to generate traction and widespread adoption of BeatBlox is the inclusion of high-quality curricula and professional development support to go along with it. We plan to generate a set of music-based curricula for BeatBlox in the coming months. In addition to standard impact and feedback metrics, a unique feature of our product is its ability to save a detailed log of all editing events made by a student, even if multiple students collaborate on a shared project from their own computers. All student work can be played back, complete with timing information, or individual events can be deduced. AI algorithms can then be used to identify the point at which students tend to struggle and provide appropriate scaffolding and identification of curriculum and tool enhancement opportunities. We plan to use this functionality to complete the feedback loop between real-time use of our tool and future development, both of the tool itself and the corresponding curriculum.

Based on interviews and feedback throughout this challenge, tools, and curricula that tick off multiple checkboxes, including addressing cross-disciplinary STEAM requirements, CS mandates, and diversity and inclusion metrics, tend to fare better in finding buy-in across the whole educational ecosystem. As such, we plan to build upon the connections we have made with students, educators, administrators, local and national organizations, and content experts to co-develop and pilot an engaging curriculum to accompany our tool and to ensure that we meet and continue to exceed the educational goals of BeatBlox.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation (grant #1949472), the NSF VITAL Prize Challenge, and a Vanderbilt Seeding Success Grant.

Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] A. Fisher and J. Margolis, “Unlocking the clubhouse: Women in computing,” in Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education,2003, p. 23.

[2] N. Z. Khan and A. Luxton-Reilly, “Is computing for social good the solution to closing the gender gap in computer science?” in Proceedings of the Australasian Computer Science Week Multiconference, 2016, pp. 1–5.

[3] S. Grover, J. Oster, A. Ledeczi, B. Broll, and M. Deweese, “Climate science, data science, and distributed computing to build teen students’ positive perceptions of cs,” in Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2, ser. SIGCSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 1101. [Online]. Available: https://doi.org/10.1145/3478432.3499079

[4] B. Broll, A. Ledeczi, P. Volgyesi, J. Sallai, M. Maroti, A. Carrillo, S. Weeden-Wright, C. Vanags, J. Swartz, and M. Lu, “A Visual Programming Environment for Learning Distributed Programming,” *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education*, pp. 81–86, 2017.

[5] B. Broll, A. Ledeczi, P. Volgyesi, J. Sallai, M. Maroti, and C. Vanags, “Introducing Parallel and Distributed Computing to K12,” *IEEE International Parallel and Distributed Processing Symposium Workshops*, pp. 323–330, 2017.

[6] J. Maloney, L. Burd, Y. Kafai, N. Rusk, B. Silverman, and M. Resnick, “Scratch: A Sneak Preview,” *Proceedings of the Second International Conference on Creating, Connecting, and Collaborating through Computing*, pp. 104–109, 2004.

[7] B. Romagosa, “The Snap*!* Programming System,” *Encyclopedia of Education and Information Technologies*, pp. 1–10, 2019.

[8] 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.

[9] B. Magerko, J. Freeman, T. Mcklin, M. Reilly, E. Livingston, S. Mccoid, and A. Crews-Brown, “Earsketch: A steam-based approach for underrep-resented populations in high school computer science education,” ACM Transactions on Computing Education (TOCE), vol. 16, no. 4, pp. 1–25, 2016

[10] H. Choi and P. Adenot, “Web Audio API,” W3C, W3C Recommendation, Jun. 2021, https://www.w3.org/TR/2021/REC-webaudio-20210617/.