USING EARSKETCH TO BROADEN PARTICIPATION IN COMPUTING AND MUSIC

Jason Freeman
Georgia Institute of Technology
jason.freeman@gatech.edu

Brian Magerko
Georgia Institute of Technology
magerko@gatech.edu

Doug Edwards
Georgia Institute of Technology
doug.edwards@ceismc.gatech.edu

Morgan Miller
SageFox Consulting
mmiller@sagefoxgroup.com

Roxanne Moore
Georgia Institute of Technology
roxanne.moore@ceismc.gatech.edu

Anna Xambó
Georgia Institute of Technology
anna.xambo@coa.gatech.edu

ABSTRACT

EarSketch is a STEAM learning intervention that combines a programming environment and API for Python and JavaScript, a digital audio workstation, an audio loop library, and a standards-aligned curriculum to teach introductory computer science together with music technology and composition. It seeks to address the imbalance in contemporary society between participation in music-making and music-listening activities and a parallel imbalance between computer usage and computer programming. It also seeks to engage a diverse population of students in an effort to address long-standing issues with underrepresentation — particularly of women — in both computing and music composition. This paper summarizes the design of the EarSketch curriculum and learning environment and its deployment contexts to date, along with key findings from a pilot study. It builds upon prior publications by contextualizing the project's motivations and interpreting its findings in the dual realms of participation in computer science and in music creation.

1. INTRODUCTION

Music has increasingly become a commodity to be heard rather than a creative experience in which to partake. Recent data from the National Endowment for the Arts in the United States (Figure 1) shows that only a small percentage of American adults engage in music-making activities even once per year, while a far greater percentage listen to recorded or live music [1].

In the field of computing, a similar divide is evident between using computers or smartphones and programming those same devices (Figure 2). Change the Equation expresses this divide succinctly in arguing that "digital native does not mean tech savvy: 83% of millennials say they sleep with their smartphones, yet 58% of millennials have poor skills in solving problems with technology" [2]. This relative lack of computational skills is more than an economic problem, with a growing demand for computing jobs in the workforce [3]. Just as music has long been a core mechanism for human expression and collaboration [4], computing is becoming a core 21st century skill: understanding the algorithms behind computers and how to write code is essential to understanding the benefits, grappling with the limitations, and harnessing the creative potential of new computing technologies [5].

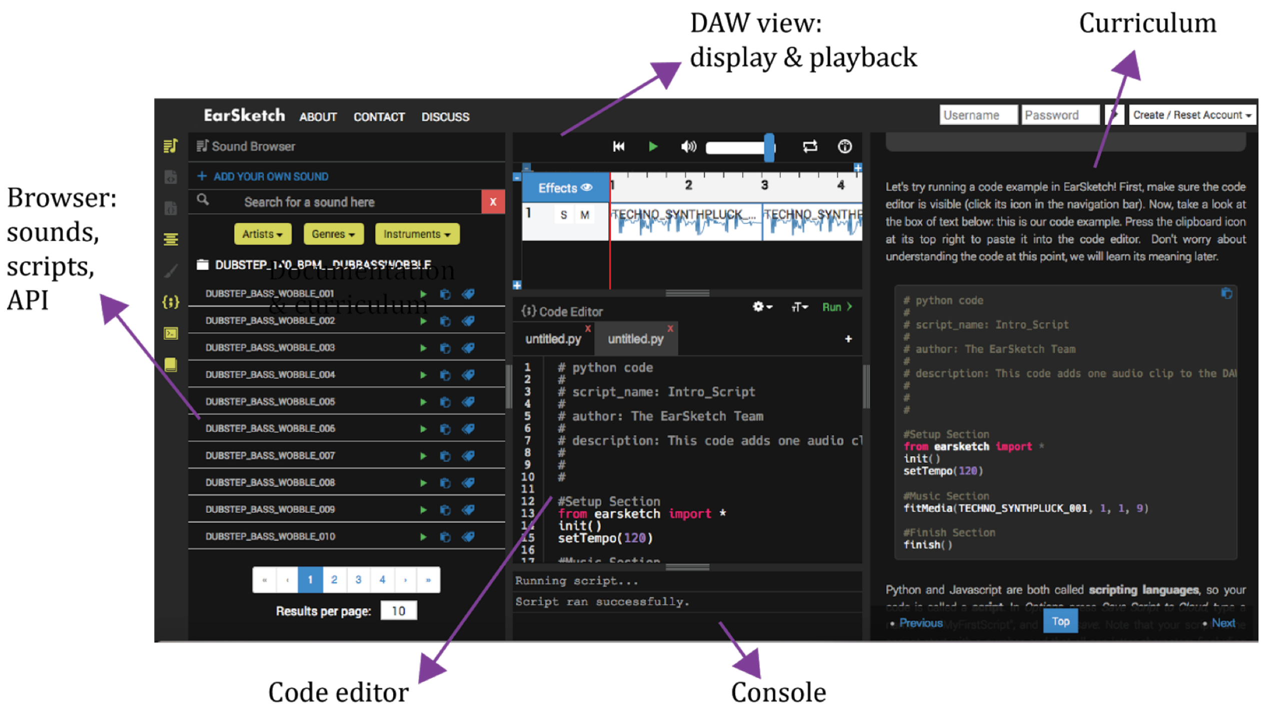
Creating Music	
Played a musical instrument, alone or with others	12%
Sang, either alone or with others	9%
Created or performed music	5%
Recorded, edited, remixed musical performances	4%
E-mailed, posted, or shared one's own music	3%
Used a computer, a handheld or mobile device, or the Internet to create music	1%
Consuming Music	
Used TV, radio, or the Internet to access music of any kind	57%
Used a handheld or mobile device to access music of any kind	34%
Attended a live music performance of any kind	32%

Figure 1. Data from the US National Endowment for the Arts on the percent of American adults (18 years and older) who have engaged in various music activities at least once over a 12-month period [1].

In the academy, computing and music have an additional commonality: both fields struggle with gender imbalance. Computer science has well-documented challenges with underrepresentation of women at all stages of the pipeline [6], with the problem generally worsening in recent decades even as other disciplines have improved [7]. Music theory and composition is one of the few academic disciplines in which an even smaller percentage of PhDs are earned by women than in computer science [8].

Coding	
"do programming" at work [9]	15%
K-12 schools offering CS courses with programming in the US [10]	25%
Students who are very likely to learn [more] computer science in the future [10]	27%
Using Computers	
Owns a desktop of laptop computer [11]	73%
Owns a smartphone of some kind [11]	68%
Uses at least one social networking site [12]	65%

Figure 2. Data on computer usage and coding.



```
# python code
#
# script_name: Intro_Script
#
# author: The EarSketch Team
#
# description: This code adds one audio cl
#
#
#

#Setup Section
from earsketch import *
init()
setTempo(120)
```

```
Running script...
Script ran successfully.
```

Let's try running a code example in EarSketch! First, make sure the code editor is visible (click its icon in the navigation bar). Now, take a look at the box of text below: this is our code example. Press the clipboard icon at its top right to paste it into the code editor. Don't worry about understanding the code at this point, we will learn its meaning later.

```
# python code
#
# script_name: Intro_Script
#
# author: The EarSketch Team
#
# description: This code adds one audio clip to the DA
#
#
#

#Setup Section
from earsketch import *
init()
setTempo(120)

#Music Section
fitMedia(TECHNO_SYNTHPLUCK_001, 1, 1, 9)

#Finish Section
finish()
```

Python and Javascript are both called **scripting languages**, so your

Figure 3. The EarSketch web-based learning environment.

It is in this context that we developed EarSketch (Figure 3), an integrated STEAM [13] programming environment, digital audio workstation, loop library, and curriculum that teaches elements of computing and music together in order to engage a diverse population of teenage students in both domains. EarSketch seeks to address the divides between music consumption and creation and between computer usage and computer programming in a manner that engages populations traditionally underrepresented in these fields.

This paper contextualizes EarSketch in related work, outlines core design principles of the project, describes the learning environment and its components in detail, summarizes the educational contexts in which it has been used, reviews results from pilot studies in schools, and outlines areas of current and future work for the project.

2. RELATED ENVIRONMENTS

EarSketch is inspired by numerous learning environments that have combined computing and music to facilitate learning in both domains through an algorithmic approach. For example, MediaComp [14] teaches introductory Python programming in part by teaching students how to implement simple audio effects. Sonic Pi [15] focuses on live coding on an embedded computing device for both sound synthesis and symbolic music generation. Performamatics [16] brings together computer science and music students to create interactive musical instruments through visual programming paradigms. And JythonMusic [17] focuses on the algorithmic generation of symbolic scores and on real-time communication with other software and devices.

In addition to these and other specialized programming environments and curricula, many popular computer music languages, such as Max [18], ChucK [19], SuperCollider [20], and Faust [21], are used pervasively in music technology pedagogy in university courses. And many programming environments designed specifically for computing education, such as Scratch [22] and PencilCode [23], include functionality for recording and playing back sound and for generating MIDI note data.

EarSketch is distinct from these other environments in three significant ways. First, EarSketch relies neither on a knowledge of symbolic music representation (e.g. MIDI note numbers or note names as in [16], [17], [18], [22], and [23]) nor on a knowledge of audio synthesis or signal processing techniques (e.g. unit generators or sample-level manipulation as in [14], [15], [18], [19], [20], and [21]).

Second, EarSketch exists primarily within the paradigm (and interface of) a digital audio workstation. Unlike Max for Live [24], the integration is not primarily through effects and plugins but rather through programmatic operations on the multi-track DAW timeline itself. In this way, EarSketch is closest in lineage to the ReaScript Python API found within the Reaper DAW [25]. (In fact, early versions of EarSketch were implemented within Reaper using ReaScript.)

Third, EarSketch focuses on enabling users to quickly create complete songs with short scripts and limited (but developing) musical and computational skills. This low barrier of entry, combined with simple design patterns to quickly create hierarchical musical structures, is meant to drive immediate engagement with music and coding for our young, novice, and diverse target audience.

3. CORE DESIGN PRINCIPLES

To further explore these distinct features of EarSketch, we now discuss five guiding design principles: creative and personal expression; real-world connections; accessibility to beginners; music-driven computational learning; and standards-based curriculum.

3.1 Creative and Personal Expression

Historically, introductory computer science has been taught as a series of abstract problems to be solved. Common student assignments ask students to sort words in a list or print out a number sequence [26].

Music, as taught to teenagers in band and orchestra classes, often focuses on rote reproduction of notated rhythms and pitches, with minimal emphasis on student creativity and expression [27]. New technologies such as SmartMusic [28] further emphasize this focus by grading students solely on how well they reproduce the notated elements of music.

EarSketch, in contrast, emphasizes open-ended assignments with no “correct” answers. Students compose music algorithmically by writing code. Their work must abide by a set of broad musical and computational constraints (e.g. to use a loop, or to incorporate at least three tracks), but students exercise wide artistic freedom and write in a wide variety of musical styles. We hope that students create music that meaningfully represents them, that they like, and that they wish to share. This approach is inspired by constructionism [29] and by studio-based learning as found in art and architecture courses [30].

3.2 Real-World Connections

To help drive student motivation, we wanted EarSketch to feel relevant to the computing and music industries.

In computing, EarSketch teaches students popular real-world programming languages. (They choose between Python or JavaScript.) This also enables students to transfer coding skills directly to other domains and contexts.

In music, EarSketch adopts the paradigm of a digital audio workstation (DAW). The user interface mimics the look and feel of popular DAWs with multi-track audio and effects lanes. The application programming interface (API) for JavaScript and Python mirrors this functionality, with core API functions to support placement of audio on the multi-track timeline, step-sequencing, and control over effects parameters and automations. The audio loop library itself provides an additional real-world connection, as the loops were created by music industry veterans: Richard Devine, an experimental electronic musician and commercial sound designer; and Young Guru, an audio engineer best known for his work with Jay-Z.

This focus on real-world connections is inspired by the notion of thick authenticity [31]. The authenticity of a learning experience, according to [32], is based on the interrelated authentic learning practices of: a) having personally meaningful learning experiences; b) learning that relates to the world outside of the learning context; c) learning that encourages thinking within a particular discipline (e.g. music composition); and d) allowing for assessment that reflects the learning process. Thick authenticity, according to [31], meets all of these requirements in a single approach / system.

3.3 Accessibility to Beginners

EarSketch was intended for widespread use amongst student and teacher populations with limited (if any) prior experience in computing or music. We therefore designed the learning environment to require no prerequisite skills in either domain.

In the computational domain, we focused our curriculum on beginning programming concepts, such as variables, functions, loops, conditionals, and lists.

In the musical domain, we teach basics of musical time and form (tempo, rhythm, measures, structures such as ABA and verse-chorus, etc.), avoid references to pitch and chord names and music notation, and structure the audio loop library as a collection of sound packs that are designed to naturally fit well together, and focus more on the hierarchical level of audio loops than on individual musical events.

3.4 Music-Driven Computational Learning

EarSketch’s grounding in digital audio workstations invites comparison to commercial music production software. There is a risk that students may be unmotivated to learn new computational concepts or to write code if they can easily achieve similar results in a traditional DAW.

We addressed this challenge by always introducing new computational concepts in service of musical ends, showing how code can sometimes create music more quickly and easily than a graphical interface, how it can enable musicians to rapidly experiment with many different musical alternatives, and how it can enable the use of musical techniques that would be impossible to achieve in a traditional DAW.

One example of this approach is the use of strings to create and vary drum beats. We introduce a string notation for step-sequencing, inspired by ixi.lang [34] and LOLC [35], in which each character represents a sixteenth-note sound hit, tie, or rest. Once these strings are created, they can be modified with string operations to be repeated, concatenated, split, shuffled, and otherwise modified. This introduces students to the notion of music as a balance between repetition and variation while providing them with the specific technique of string creation and manipulation to actualize this concept in music.

3.5 Standards-Based Curriculum

To facilitate widespread adoption of EarSketch in learning contexts aimed at our target age demographic, we focused on high-school computer science classrooms and on a new curriculum standard in the United States: Computer Science Principles [33].

Our curriculum, and by extension core features of the EarSketch API and learning environment, were therefore designed specifically to address the learning objectives in the Computer Science Principles framework. For example, EarSketch focuses primarily on imperative programming paradigms and on constructs for iteration, abstraction, and branching that fit within such paradigms, while avoiding object-oriented structures or functional approaches that, while used widely in music technology (e.g. [20]), are not emphasized in the Computer Science Principles framework.

4. THE LEARNING ENVIRONMENT

EarSketch is a free, web-based environment that integrates multiple components within a single browser window [36]. In this section, we describe its main components: the programming environment, the digital audio workstation (DAW), the loop library, and the curriculum.

4.1 Programming Environment

In the EarSketch code editor, students write code in Python or JavaScript, using either a text editor or a blocks-based visual code editor [37]. Regardless of language or editor chosen, they use the same application programming interface (API) to create music.

Figure 4 shows a simple EarSketch program. `fitMedia()` places an audio clip on a particular track and starting/ending times, looping or truncating the clip as necessary to fill the specified amount of time. `makeBeat()` step-sequences a rhythm, with each character of a string representing a sixteenth-note: a "0" plays a sound file from the beginning, another digit plays alternate sound files at other indices within a list, a "+" ties (or continues playing) the sound file, and a "-" makes a rest (silence). `setEffect()` adds an effect to a track (or the master track), with optional parameters to specify effect parameters and to define an envelope for those parameters.

```
from earsketch import *

init()
setTempo(120)

fitMedia(HOUSE_ROADS_PIANO_007, 1, 1, 3)
setEffect(1, VOLUME, GAIN, -60, 1, 0, 3)

beatElement = OS_LOWTOM01
beatString = "0+++0+++0+0+0+0+"
for index in range(1,3):
   makeBeat(beatElement,2,index,beatString)

finish()
```

Figure 4. A sample EarSketch Python script that places an audio clip on track 1, adds a volume effect automation to track 1, and places a step-sequenced beat on track 2.

Figure 5 shows a more complex example that mimics the practice of hocketing to create a hybrid drum track out of two audio sources. For each sixteenth note in the timeline, the RMS amplitude of each track is computed. The louder track's level is then set to 0 dB for that sixteenth note, and the quieter track's level is set to -60 dB.

Additional API methods offer alternate methodologies to audio file placement and implement utility functions such as console and file input and string manipulation.

By default, EarSketch operates in a batch mode. Code is interpreted when hitting the "run" button to create the music. It does not run interactively while the music is playing. This approach follows a music production methodology which is focused more on creating a fixed-media track than on live performance. EarSketch does support live coding [38]. Users can write and execute code while audio is playing, and audio playback will update seamlessly.

```
from earsketch import *
init()
setTempo(120)

sound1 = ELECTRO_DRUM_MAIN_BEAT_001
sound2 = ELECTRO_DRUM_MAIN_BEAT_002
analysisMethod = RMS_AMPLITUDE
hop = 0.0625 # analyze 1/16th note chunks
start = 1
end = 3.0

fitMedia(sound1, 1, start, end)
fitMedia(sound2, 2, start, end)

position = 1
while (position < end):
   feature1 = analyzeTrackForTime(1,
      analysisMethod, position,
      position + hop)
   feature2 = analyzeTrackForTime(2,
      analysisMethod, position,
      position + hop)
   if (feature1 > feature2):
      setEffect(1, VOLUME, GAIN, 0,
         position, 0, position + hop)
      setEffect(2, VOLUME, GAIN, -60,
         position, -60, position + hop)
   else:
      setEffect(1, VOLUME, GAIN, -60,
         position, -60, position + hop)
      setEffect(2, VOLUME, GAIN, 0,
         position, 0, position + hop)
   position = position + hop

finish()
```

Figure 5. An algorithmic EarSketch example in which two tracks' mute states are toggled every sixteenth note depending on which has the higher amplitude.

4.2 Digital Audio Workstation

The digital audio workstation panel within EarSketch displays the visual output of code execution in a standard multi-track format. It is not a fully-functional DAW in that students cannot add, edit, or delete audio clips or effects; this must be done through coding. Students can navigate their project by using transport controls, soloing/muting tracks, and bypassing effects. They can also export their project as a stereo mixdown (WAV, MP3, or Soundcloud) or a multi-track project to continue editing in a traditional desktop DAW.

4.3 Loop Library

EarSketch includes ~4000 loops accessible via a sound browser sidebar. The sound browser pane mimics the functionality of similar interface panels in DAWs, allowing users to search and filter sounds by artist, genre, and instrument. Sounds are grouped into collections that contain loops designed to fit well together. Users may also upload their own sounds from their computer or quick-record new sounds directly within EarSketch.

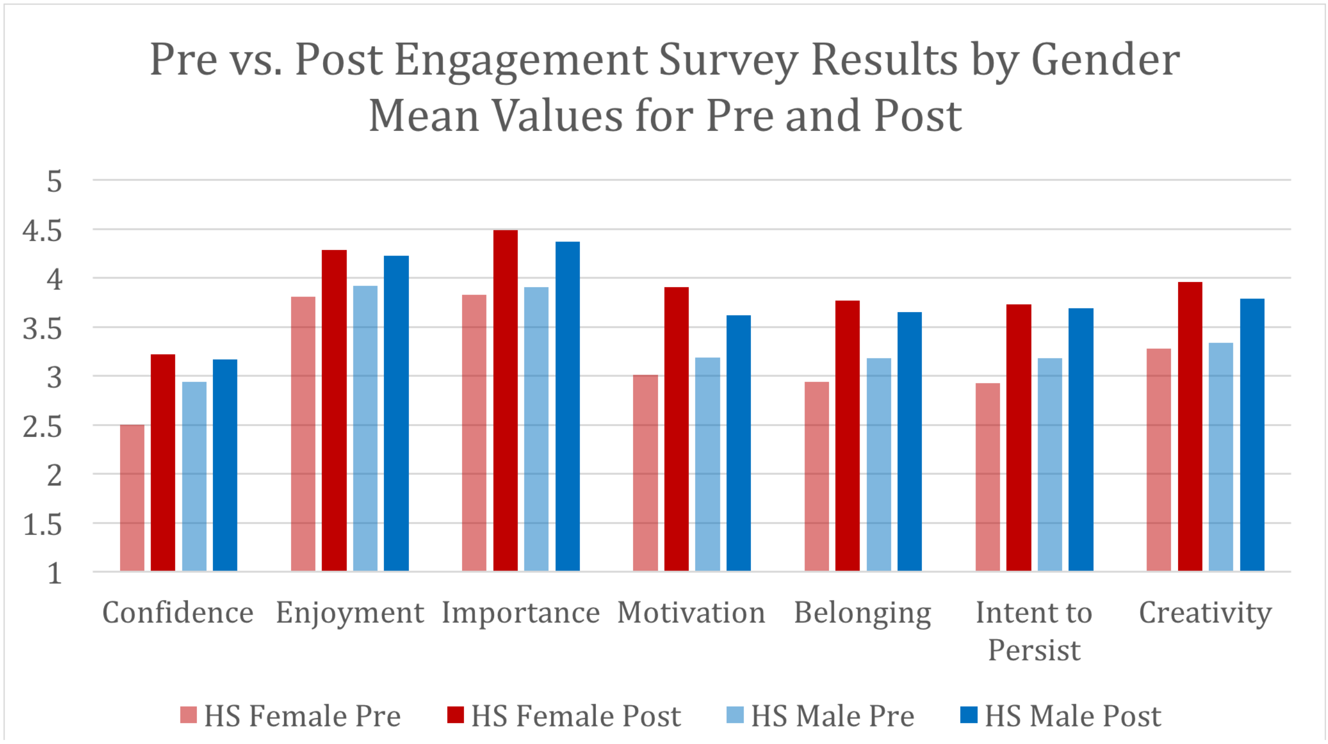


Figure 6. Pre and post engagement survey results across male and female students from a 2013 EarSketch pilot study. Across all seven engagement constructs, female students are less engaged at pre than their male counterparts but more engaged at post than the male students.

Each sound in the library is identified by a unique constant. To use the sounds within EarSketch, users simply paste the constants into the code editor as function arguments. EarSketch automatically time-stretches loops to match the overall project tempo.

4.4 Curriculum

The EarSketch curriculum is intended to be used within introductory computing and music technology courses, and is specifically aligned to AP Computer Science Principles [33], an emerging curriculum standard in the United States for computer science courses at the high school (teenage) level. The EarSketch curriculum covers computational topics such as data types, variables, functions, lists, loops, boolean logic, conditionals, and strings, and music and music technology concepts such as DAW basics, musical form, rhythm, meter, tempo, and texture.

A sidebar within EarSketch displays textbook-like materials for students to use for self-study and as a reference: this includes text, runnable code examples, video demonstrations, and slides. Classroom instructors can access teaching materials that include day-by-day lesson plans, handouts, and projects and assessments.

Each summer since 2014, the EarSketch team has conducted professional learning workshops for teachers interested in adopting EarSketch. These workshops teach EarSketch and the music and computing fundamentals teachers need for the course, as well as pedagogical techniques on topics such as facilitating student collaboration, discussing student projects, and assisting students in debugging code.

5. DEPLOYMENT CONTEXTS

EarSketch has been used in a variety of educational contexts, including academic courses in computing and in music technology at high schools; summer camps for middle school and high school students; undergraduate-level introductory computing courses; and a Massive Open Online Course (MOOC) in music technology taught by one of the authors (Freeman) on Coursera [39].

Since we launched the web based version of EarSketch in 2014, over 55,000 unique users from over 100 countries have coded with EarSketch, saving over 47,000 projects to our server.

6. PILOT RESULTS

Between 2013 and 2015, we have conducted multiple pilot studies of EarSketch in academic computing courses at four different Atlanta-area high schools. To study the impact of EarSketch on students in these courses, we employed a variety of research methods, including questionnaires, content assessments, observations, interviews, and focus groups. The content knowledge assessment, given before and after the EarSketch module of the course, was a multiple-choice assessment aligned to the learning objectives of the courses.

A student engagement survey, administered retrospectively, monitored potential changes in students' internal characteristics. This instrument draws scales from Williams, Weibe, Yang, & Miller [40] and Knezek & Christensen [41] measuring computing confidence, computing enjoyment, computing importance and perceived usefulness, motivation to succeed, and computing identity and belongingness as predictor variables and an intention to persist in computing as an outcome variable. The literature in STEM education suggests that these constructs are critical to enhancing the number of under-represented students who persist in STEM fields [40]. Earlier versions of the instrument also adopted the Creativity Support Index [42], but more recently we have developed our own questions to gauge students' perceptions about creativity, building on prior research [43], [44] on creativity.

We now summarize findings of our 2013 pilot study [45], in which we compared results of male and female EarSketch students. The study included students in two courses. One was an introductory computing course; the other was an introductory music technology course. Both included a similar EarSketch curricular module. 97 students provided usable data across all student survey constructs, with 27% of them female and 73% male (a typical breakdown for these courses). Students did not know they would be using EarSketch prior to course enrollment.

Both male and female students showed statistically significant increases from pre to post across all engagement constructs ($p < 0.01$), with the exception of male confidence ($p = 0.07$). Furthermore, female students expressed greater pre-to-post change across all constructs than male students; these differences are significant ($p < 0.05$) in confidence, motivation, and identity and belongingness. Figure 6 shows this visually: across all constructs, female students are less engaged than their male counterparts before they study EarSketch but are more engaged than the male students after studying EarSketch. Both male and female students' content knowledge also significantly increased from pre to post, but there were no significant differences between male and female student gains in this area. These results are discussed in more detail in [45].

Focus groups and free-response items in student surveys suggest that the core design principles guiding EarSketch play an important role in student engagement. Some students remarked on the importance of personal expression and creativity, commenting that "I got to express my ideas and it was fun and inspiring to see that I could be good at computing" and "I enjoyed making my own music tracks that people, including myself, actually liked." Others focused on the importance of real-world context, noting that "I liked learning how music is made and how we can learn and get good at doing things that people in the music industry do now." This seemed to impact students' interest in persisting in further study, as evidenced by this comment from a focus group: "It gives me choices for college. Like this is something I would actually like to do for college and I'd actually like to do probably with my life. Yeah. I would love to do it."

Our pilot studies have been more focused on computer science content knowledge and attitudes than on music, but in a fall 2015 pilot study, we did collect data on students' experiences with music prior to the course. 83% of students stated that they listened to music for at least one hour every day. Only 6% of students had mixed or composed their own music prior to using EarSketch, and only 36% were involved in music performance activities such as band, chorus, orchestra, or instrumental lessons. This imbalance between music listening and music making mirrors the data from the US National Endowment for the Arts. In future studies we hope to measure if and how EarSketch has engaged students in music making beyond the course and beyond EarSketch itself.

These pilot results suggest that EarSketch has strong potential to engage students — and particularly female students — in computing and music at the introductory level.

7. DISCUSSION AND FUTURE WORK

Adoption of EarSketch is growing rapidly, and we are currently scaling up our research efforts to understand how EarSketch impacts student engagement and content knowledge across diverse populations and school contexts. Over the next two years, we will be expanding our research efforts to study EarSketch in approximately 30 high school AP Computer Science Principles classrooms in Georgia, using content knowledge assessments, engagement surveys, observations, interviews, and focus groups to understand how EarSketch impacts students and how we can continue to improve it. As part of this study, we are also comparing classrooms using EarSketch to classrooms using other learning environments. We are also using complex systems modeling techniques [46] to model the complex sets of attributes and relationships that underlie learning interventions.

At the same time, we are expanding EarSketch to new modalities and to new learning contexts. We are continuing to develop a blocks-based visual programming editor for EarSketch that will enable us to more successfully incorporate it into classrooms with younger students. We are also currently developing a collaborative, tabletop interface suited for museum installations and outreach events. We recently added support for P5 [47] into EarSketch to support the generation of live visuals alongside music, and are exploring ways to connect EarSketch to physical computing systems such as the Moog Werkstatt [48] and the Lilypad Arduino [49]. We are also interested in finding ways to integrate EarSketch into other computer music and general-purpose programming environments.

Regardless of the modality and context, our goals remain the same: to engage a broad and diverse population in making music and writing code, and in doing so to spark their interest in these activities such that they persist and continue to develop beyond a single learning intervention.

Acknowledgements

The EarSketch project receives funding from the National Science Foundation (CNS #1138469, DRL #1417835, and DUE #1504293), the Scott Hudgens Family Foundation, the Arthur M. Blank Family Foundation, and the Google Inc. Fund of Tides Foundation. EarSketch is available online at http://earsketch.gatech.edu. A com-

plete list of personnel on the EarSketch team is available at http://earsketch.gatech.edu/landing/team2.html.

8. REFERENCES

[1] S. Iyengar, "How a Nation Engages with Art: Highlights from the 2012 Survey of Publication Participation in the Arts," National Endowment for the Arts, 57, 2013.

[2] Change the Equation, "Digital Native Does Not Mean Tech Savvy," 11-Jun-2015. [Online]. Available: http://changetheequation.org/stemtistics/digital-native-does-not-mean-tech-savvy. [Accessed: 25-Mar-2016].

[3] Code.org, "Promote Computer Science," *Code.org*. [Online]. Available: https://code.org/promote. [Accessed: 25-Mar-2016].

[4] I. Peretz, "The nature of music from a biological perspective," *Cognition*, vol. 100, no. 1, pp. 1–32, May 2006.

[5] J. M. Wing, "Computational thinking," *Commun ACM*, vol. 49, no. 3, pp. 33–35, 2006.

[6] J. Margolis, "Unlocking the Clubhouse: A Decade Later and Now What?," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, New York, NY, USA, 2013, pp. 9–10.

[7] National Science Foundation, "Science and Engineering Degrees: 1996-2010," 2013. [Online]. Available: http://www.nsf.gov/statistics/nsf13327/content.cfm?pub_id=4266&id=2. [Accessed: 25-Mar-2016].

[8] S.-J. Leslie, A. Cimpian, M. Meyer, and E. Freeland, "Expectations of brilliance underlie gender distributions across academic disciplines," *Science*, vol. 347, no. 6219, pp. 262–265, Jan. 2015.

[9] C. Scaffidi, M. Shaw, and B. Myers, "Estimating the numbers of end users and end user programmers," in *2005 IEEE Symposium on Visual Languages and Human-Centric Computing*, 2005, pp. 207–214.

[10] Gallup, "Images of Computer Science: Perceptions Among Students, Parents, and Educators in the U.S.," Google, 2015.

[11] M. Anderson, "Technology Device Ownership: 2015," *Pew Research Center: Internet, Science & Tech*, 29-Oct-2015. .

[12] A. Perrin, "Social Media Usage: 2005-2015," *Pew Research Center: Internet, Science & Tech*, 08-Oct-2015. .

[13] J. Maeda, "STEM+ Art = STEAM," *STEAM J.*, vol. 1, no. 1, p. 34, 2013.

[14] M. Guzdial, "A media computation course for non-majors," *ACM SIGCSE Bull.*, vol. 35, pp. 104–108, 2003.

[15] S. Aaron and A. F. Blackwell, "From Sonic Pi to Overtone: Creative Musical Experiences with Domain-specific and Functional Languages," in *Proceedings of the First ACM SIGPLAN Workshop on Functional Art, Music, Modeling &#38; Design*, New York, NY, USA, 2013, pp. 35–46.

[16] J. M. Heines, G. R. Greher, and S. Kuhn, "Music Performamatics: Interdisciplinary Interaction," in *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, New York, NY, USA, 2009, pp. 478–482.

[17] B. Manaris and A. R. Brown, *Making Music with Computers: Creative Programming in Python*, vol. 13. Boca Raton, FL: CRC Press, 2014.

[18] M. Puckette, "Combining event and signal processing in the MAX graphical programming environment," *Comput. Music J.*, pp. 68–77, 1991.

[19] G. Wang, P. R. Cook, and S. Salazar, "ChucK: A Strongly Timed Computer Music Language," *Comput. Music J.*, vol. 39, no. 4, pp. 10–29, Dec. 2015.

[20] J. McCartney, "Rethinking the Computer Music Language: SuperCollider," *Comput. Music J.*, vol. 26, no. 4, pp. 61–68, Dec. 2002.

[21] Y. Orlarey, D. Fober, and S. Letz, "Syntactical and semantical aspects of Faust," *Soft Comput.*, vol. 8, no. 9, pp. 623–632, Jul. 2004.

[22] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, and B. Silverman, "Scratch: programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, 2009.

[23] D. Bau, D. A. Bau, M. Dawson, and C. Pickens, "Pencil code: block code for a text world," in *Proceedings of the 14th International Conference on Interaction Design and Children*, 2015, pp. 445–448.

[24] V. J. Manzo and W. Kuhn, *Interactive Composition: Strategies Using Ableton Live and Max for Live*. Oxford: Oxford University Press, 2015.

[25] J. Frankel, "ReaScript," 2005. [Online]. Available: http://www.reaper.fm/sdk/reascript/reascript.php. [Accessed: 01-Jan-2015].

[26] L. Layman, L. Williams, and K. Slaten, “Note to self: make assignments meaningful,” *ACM SIGCSE Bull.*, vol. 39, no. 1, pp. 459–463, 2007.

[27] R. E. Allsup, “Mutual learning and democratic action in instrumental music education,” *J. Res. Music Educ.*, vol. 51, no. 1, pp. 24–37, 2003.

[28] R. Gurley, “Student perception of the effectiveness of SmartMusic as a practice and assessment tool on middle school and high school band students,” Texas Tech University, 2012.

[29] Y. Kafai, “Constructionism,” in *Cambridge Handbook of the Learning Sciences*, K. Sawyer, Ed. Cambridge, MA: Cambridge University Press, 2006, pp. 35–46.

[30] L. Hetland, L. Winner, S. Veenema, and K. Sheridan, *Studio Thinking: The Real Benefits of Arts Education*. New York, NY: Teachers College Press, 2007.

[31] D. W. Shaffer and M. Resnick, “‘ Thick’ Authenticity: New Media and Authentic Learning.,” *J. Interact. Learn. Res.*, vol. 10, no. 2, pp. 195–215, 1999.

[32] H.-S. Lee and N. Butler, “Making authentic science accessible to students - International Journal of Science Education,” *Int. J. Sci. Educ.*, vol. 25, no. 8, pp. 923–948, 2003.

[33] O. Astrachan and A. Briggs, “The CS Principles Project,” *ACM Inroads*, vol. 3, no. 2, pp. 38–42, Jun. 2012.

[34] T. Magnusson, “ixi lang: A SuperCollider Parasite for Live Coding,” in *SuperCollider Symposium 2010*, Berlin, 2010.

[35] J. Freeman and A. V. Troyer, “Collaborative Textual Improvisation in a Laptop Ensemble,” *Comput. Music J.*, vol. 35, no. 2, pp. 8–21, May 2011.

[36] A. Mahadevan, J. Freeman, B. Magerko, and J. C. Martinez, “EarSketch: Teaching computational music remixing in an online Web Audio based learning environment,” in *Proceedings of the 1st Annual Web Audio Conference*, Paris, 2015.

[37] D. Bau, “Droplet, a blocks-based editor for text code,” *J. Comput. Sci. Coll.*, vol. 30, no. 6, pp. 138–144, 2015.

[38] J. Freeman and B. Magerko, “Iterative composition, coding and pedagogy: A case study in live coding with EarSketch,” *J. Music Technol. Educ.*, vol. 9, no. 1, 2016.

[39] J. Freeman, “Survey of Music Technology,” *Coursera*, 2015. [Online]. Available: https://www.coursera.org/learn/music-technology. [Accessed: 25-Mar-2016].

[40] E. Wiebe, L. Williams, K. Yang, and C. Miller, “Computer science attitude survey,” *Comput. Sci.*, vol. 14, no. 25, 2003.

[41] G. Knezek and R. Christensen, “Validating the Computer Attitude Questionnaire (CAQ).,” 1996. [Online]. Available: http://files.eric.ed.gov/fulltext/ED398243.pdf.

[42] E. A. Carroll, C. Latulipe, R. Fung, and M. Terry, “Creativity factor evaluation: towards a standardized survey metric for creativity support,” in *Proceedings of the seventh ACM conference on Creativity and cognition*, 2009, pp. 127–136.

[43] T. M. Amabile, “Within you, without you: The social psychology of creativity, and beyond,” *Theor. Creat.*, vol. 4, pp. 61–91, 1990.

[44] R. E. Mayer, “22 Fifty Years of Creativity Research,” *Handb. Creat.*, vol. 449, 1999.

[45] 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 Trans. Comput. Educ.*, in press 2016.

[46] D. Llewellyn, M. Usselman, D. Edwards, R. Moore, and P. Mital, “Analyzing K-12 Education as a Complex System,” in *Proceedings of the ASEE 2013 Annual Conference*, 2013.

[47] L. McCarthy, “p5.js,” 2016. [Online]. Available: http://p5js.org. [Accessed: 26-Mar-2016].

[48] C. Howe, “Analog Synthesizers in the Classroom,” Georgia Institute of Technology, Atlanta, GA, 2014.

[49] L. Buechley, M. Eisenberg, J. Catchen, and A. Crockett, “The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2008, pp. 423–432.