TunePad: Computational Thinking Through Sound Composition

Jamie Gorson
Computer Science and Learning Sciences
Northwestern University
Evanston, IL, 60208 USA
jgorson@u.northwestern.edu

Nikita Patel
Computer Science
Northwestern University
Evanston, IL, 60208 USA
nikitapatel@u.northwestern.edu

Elham Beheshti
Computer Science
Northwestern University
Evanston, IL, 60208 USA
beheshti@u.northwestern.edu

Brian Magerko
Digital Media Program
Georgia Institute of Technology
Atlanta, GA 30332 USA
magerko@gatech.edu

Michael Horn
Computer Science
Northwestern University
Evanston, IL, 60208 USA
michael-horn@northwestern.edu

IDC '17, June 27-30, 2017, Stanford, CA, USA

ACM ISBN 978-1-4503-4921-5/17/06.
http://dx.doi.org/10.1145/3078072.3084313

Abstract
Computational thinking skills will be important for the next generation of students. However, there is a disparity in the populations joining the field. Integrating computational thinking into artistic fields has shown to increase participation in computer science. In this paper, we present our initial design prototype for TunePad, a sound composition tablet application controlled by a block-based programming environment. TunePad is designed to introduce learners to computational thinking and to prepare them for text-based coding environments. From our preliminary testing, with children ages 7-14, we observed that our design actively engages learners and communicates how the programming blocks control the sounds being played. This testing is a prelude to more formal studies as we continue to improve the design and interface of TunePad. With this work, we intend to engage students in computational thinking who may not have otherwise been exposed, giving the opportunity to more people to enter the computer science field.

Author Keywords
Computational thinking; music composition; broadening participation; CS education; STEAM;

ACM Classification Keywords
H.5.2 [User Interfaces]: User-Centered Design; K.3.2 [Computer and Information Science Education]:

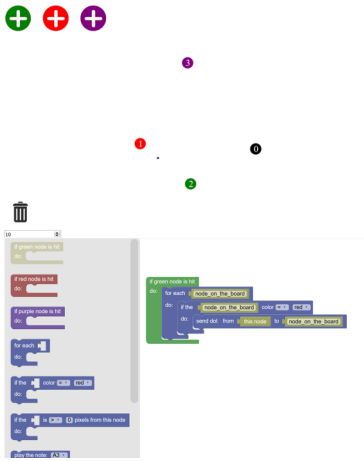

Figure 1: Screenshot of the TunePad prototype.

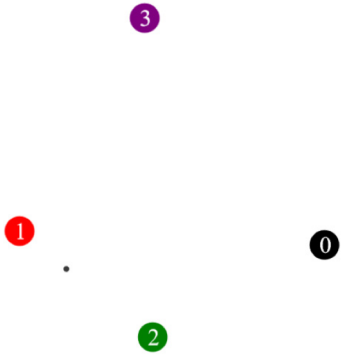

Figure 2: Nodes, generator and particle on the TunePad prototype.

Computer Science Education; J.5 [Arts and Humanities]: Music;

Introduction

Computational thinking skills are becoming increasingly important for the next generation [13]. However, large disparities remain in populations pursuing computer science, in part because many students are turned off by the image portrayed by the field [4]. One promising approach to address this disparity has been STEAM (Science, Technology, Engineering, Arts, Mathematics), the integration of the arts into technology-focused fields like computing [9]. STEAM blends topics perceived as "rigid and logical-mathematical" with disciplines seen as more "creative" and broadly appealing [7].

One example of the STEAM based approach is EarSketch, an online learning environment that integrates computer programming with electronic music composition. With EarSketch, learners from both minority and majority populations have shown significant increases in content knowledge, along with positive attitudinal shifts towards computing [6]. EarSketch lets students remix professionally produced audio samples with authentic programming languages (Python or JavaScript). However, EarSketch's curriculum and text-based programming environment are targeted at students in high school and may be intimidating and difficult for younger learners.

Building on EarSketch, we are designing and testing a sound composition app called TunePad. Our goal is to provide a playful introduction to computational thinking in places like community centers, libraries, schools, and homes. Learners manipulate a visual interface consisting of colorful nodes that generate sounds. Users can program these nodes with a visual programming environment based on the Blockly framework [5]. The application is ideally easy enough for young learners to pick up in informal, unguided contexts, but rich enough to help them transition to more sophisticated environments over time. With TunePad, we hope to broaden participation in computing.

Our overarching research questions are (1) how can we best design a learning environment that engages novices in computational thinking around music composition and (2) how do we strike a balance between playful, informal learning while also preparing learners for transferring to a more instruction-based learning environment such as EarSketch?

In this paper, we describe our prototype design and present findings from our preliminary testing with 26 children. From these sessions, we found that our design was engaging for learners. Our participants, many of whom had prior experience with block-based programming, understood how TunePad programming blocks connected to nodes. We still have open questions about how to help users learn computational theory through our coding blocks without sacrificing the playful and exploratory nature of the experience.

Background

Many researchers have studied environments to help children learn computational thinking concepts and change their attitudes towards computer science [3,4,11]. The STEAM approach has been promising in its ability to improve personal motivation and increase learner engagement [7]. For example, Buechly et al. showed that using e-textiles to create wearables was

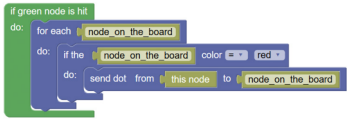

Figure 3: Example of a reaction function.

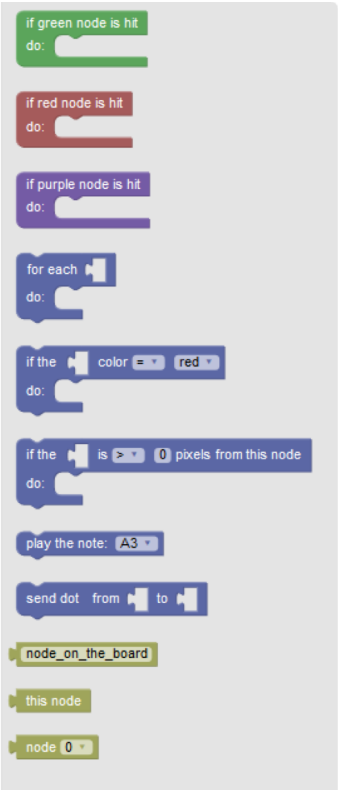

Figure 4: Custom-made blocks for TunePad.

an engaging and rich vehicle for teaching computational thinking to children [3]. STEAM has also been applied to music composition, integrating music with STEM fields [1,2,6]. This strategy takes advantage of pre-existing communities and interests. These studies found that learners engage in STEM, and specifically computing, when it's integrated into the learner's existing activities. We hope to follow this strategy by engaging learners who are already drawn to music while introducing them to computational thinking.

Prototype Design

TunePad (Figure 2) is comprised of nodes (colored dots), generators (black dot) and particles (small grey dots), which enable learners to create sound. In TunePad, learners can experiment by placing nodes on the screen and observing the resulting sounds created when particles hit the nodes. We were inspired by an existing music app called NodeBeat, in which users create combinations of pitches and sounds to rapidly produce musical rhythms [12].TunePad is distinct from NodeBeat because by using block code, learners can tinker with a node's actions. The generator sends a particle to every node at regular intervals. When a particle hits a node, the learner now determines which sounds are created and if new particles are sent from nodes by writing the reaction function.

Users program the reaction function (Figure 3) using our custom-made block-based programming language created with the Blockly framework [5]. The method of programming is based on the object-oriented programing model. This means that the code is organized around objects instead of actions. In our implementation, the code is oriented around the node that is hit. Nodes are assigned a number based on their order added to the board, allowing users to identify and target individual nodes. Reaction functions are written for each color of node, meaning all nodes of the same color execute identical code.

There are 12 different programming blocks in the prototype (Figure 4). The first three are the containers for each reaction function and colored to match their respective nodes. The middle five, all navy blue, are the command blocks. The 'for each' block allows users to access all the nodes on the board. The conditionals afford the user control over where to send particles based on color or distance. This gives the user more creative ability and teaches them about if statements. The play block is what makes the sounds. The send dot block creates a particle from the first input to the second input, teaching participants about calling functions with parameters. The last three blocks, all yellow, are how users identify nodes. The first is a variable to be used with the 'for each' block. The 'this node' block calls the node that was hit by the particle. The final block has a dropdown where users can pick an individual node to reference by its number.

The text on each block was written so that a complete function could be read and comprehended with natural language, yet formatted in a way that correlates to programming syntax. This way, a user could easily understand the blocks and also learn computing concepts.

To help children learn the fundamental ideas of troubleshooting and debugging, we have begun to incorporate error messages into the program. For example, if a user accumulates too many particles on the board by creating an infinite loop, the user will

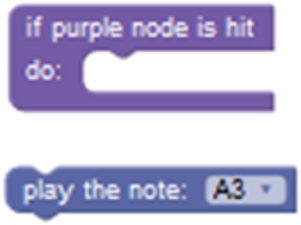

Figure 5: A function block and a command block.

receive a notification that the song exploded. When this happens, all the particles are erased on the board leaving the code and nodes the same, forcing the user to correct their work to avoid another explosion.

Prototype Testing

We tested our prototype with 14 children, ages 7–14, in an after-school music school in a major Midwest city. We asked children to use the tablet in groups of two or three. Two of the participants used the application alone. Each group was monitored by a researcher who helped and observed the group. We tested on two separate days with different children each time. On the first day, we provided a printed example of a working program for the students to replicate, and then encouraged each group to experiment freely. On the second day, we provided the learners with a printed set of step-by-step instructions to teach them how to create a working program. At the end of each session, the researcher asked the children for feedback on their experience with the application.

Additionally, we tested the prototype with 12 children, ages 10-14, during a workshop in a STEAM-based college-prep program. We asked the children to use the prototype in groups of three for 7 minutes. The students were encouraged to explore on their own, with minimal instruction from the researcher, who only stepped in to help when a child demonstrated signs of lost interest or asked for assistance. After finishing interacting with the app, researchers asked the participants for feedback.

Preliminary Findings and Discussion

Engagement

Almost all participants expressed interest in using TunePad beyond the allocated study time and most were physically and vocally enthusiastic while using the app. Many groups, after making an arrangement of nodes and blocks, watched their results with big smiles on their faces. Some of the groups increased the number of particles so much that the sound got very loud. One participant put his hands up to his ears and commented "Ahh! It's horrible again!" referring to the explosion of noise. This dislike encouraged him to edit the code to improve his composition. We expected learners to spend around 10 minutes playing with the app, but most groups interacted for around 30 minutes, even after researchers asked if they wanted to stop. Some children didn't want to leave and asked how they could play it at home.

Understanding Programming Controls Nodes

The groups showed a compelling understanding of how manipulating the programming blocks controlled the nodes on the screen. They would move the blocks and look for changes in the particles and sounds. While they were almost always able to reproduce the examples given, they were not able to fully explain each line of code. Students seemed to understand the mapping between a pattern of code blocks and a desired result, but struggled to create new patterns. The groups varied drastically in age and prior coding experiences, resulting in a range of the complexity of their compositions and their understanding of the computational concepts.

Intuitiveness of Block-based Programming

Many of our participants had previously exposure to block-based programming, so they easily understood the interaction. What we found least intuitive about

using the blocks for musical programming was indicating which blocks needed to be inside a function. The command blocks have indents on the top and a carrot on the bottom to imply that blocks go before and after, while the function blocks have the indent on the inside, so users put blocks on the inside (Figure 5). Several participants put the command blocks on the workspace without putting them into a function block and expected it to have an effect. Since reaction functions run when a node is hit, these independent blocks will never execute. We hoped the Blockly notation would discourage this, but it was not apparent to the participants.

Comprehension and Interest Without Instruction
At the second site, the groups were not given direction on how to use the app, yet all but one of the groups were able to create a composition. While these students struggled slightly more than those given direction, they figured out how to use the application to create sound. Their interest level in the application remained high until they couldn't understand an individual coding block or became confused.

Future Work

Develop In-App Directions
We are still very unsure how to best introduce TunePad in a way that helps children learn computational thinking without sacrificing the playful and exploratory nature of the application. While we tried prototypes of two methods in the preliminary testing, we still have limited understanding. In further testing we hope to formatively test different models of instruction.

Improving Functionality
We want to build upon the current functionality of the application, giving users more control over the sounds and adding more computational thinking elements. For example, nodes could be chosen by properties of their number, like evens and odds and sounds could be played in different instruments or octaves. Increasing the possible variables allows for the integration of more computational theory, like Boolean logic. With more complexity, we would need a method for revealing tools slowly to not overwhelm the learner.

Tangibles
We are also interested in studying the effects of adding physical manipulatives (tangibles) to the digital interface. Research has shown that physical manipulatives can support mathematics learning [6] or can increase engagement [8]. These tangibles could be as simple as print-at-home representations of the nodes, ensuring that the addition of tangibles wouldn't decrease the accessibility of the application.

Conclusion

We presented the initial prototype design of TunePad, a sound composition app that introduces computational thinking skills to middle-school aged learners. Although we have only conducted preliminary testing, we found that TunePad is engaging, making it a potential platform for introducing learners to computational thinking skills. The continuation of this work and future studies will help inform designers using STEAM approaches and create a new pathway for children to computer science. TunePad has shown to attract music students to computational thinking and we hope it will help attract a broader population to computing fields.

Acknowledgements

We thank our participants and Anna Xambo for help with this study. We thank the National Science Foundation for their support of this project (DRL-1612619). 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 National Science Foundation.

References

1. Samuel Aaron and Alan F. Blackwell. 2013. 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 & design*, 35–46.
2. Jeanne Bamberger. 2003. Music as embodied mathematics: A study of a mutually informing affinity. *International Journal of Computers for Mathematical Learning* 8, 2: 123–160.
3. Leah Buechley, Mike Eisenberg, Jaime Catchen, and Ali Crockett. 2008. 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*, 423–432. Retrieved March 7, 2017 from http://dl.acm.org/citation.cfm?id=1357123
4. Jamika D. Burge and Tiki L. Suarez. 2005. Preliminary analysis of factors affecting women and African Americans in the computing sciences. In *Proceedings of the 2005 conference on Diversity in computing*, 53–56. Retrieved March 12, 2017 from http://dl.acm.org/citation.cfm?id=1095265
5. N. Fraser. 2013. Blockly: A visual programming editor. *Published. Google, Place*.
6. Jason Freeman, Brian Magerko, Tom McKlin, Mike Reilly, Justin Permar, Cameron Summers, and Eric Fruchter. 2014. Engaging Underrepresented Groups in High School Introductory Computing Through Computational Remixing with EarSketch. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (SIGCSE '14), 85–90. https://doi.org/10.1145/2538862.2538906
7. Danah Henriksen. 2014. Full STEAM Ahead: Creativity in Excellent STEM Teaching Practices. *STEAM* 1, 2: 1–9. https://doi.org/10.5642/steam.20140102.15
8. Michael S. Horn, Erin Treacy Solovey, R. Jordan Crouser, and Robert JK Jacob. 2009. Comparing the use of tangible and graphical programming languages for informal science education. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 975–984. Retrieved October 27, 2016 from http://dl.acm.org/citation.cfm?id=1518851
9. John Maeda. 2013. STEM + Art = STEAM. *STEAM* 1, 1: 1–3. https://doi.org/10.5642/steam.201301.34
10. Taylor Martin and Daniel L. Schwartz. 2005. Physically distributed learning: Adapting and reinterpreting physical environments in the development of fraction concepts. *Cognitive science* 29, 4: 587–625.
11. Andrés Monroy-Hernández and Mitchel Resnick. 2008. Empowering Kids to Create and Share Programmable Media. *interactions* 15, 2: 50. https://doi.org/10.1145/1340961.1340974
12. S. Sandler and J. Windl. 2013. NodeBeat. app for iOS. *Blackberry, Android, and Windows*.
13. David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. 2016. Defining Computational Thinking for

Mathematics and Science Classrooms. *Journal of Science Education and Technology* 25, 1: 127–147. https://doi.org/10.1007/s10956-015-9581-5