Considering Alternative Endpoints: An Exploration in the Space of Computing Educations

David Weintrop, *weintrop@umd.edu*
College of Education & College of Information Studies,
University of Maryland, College Park, USA

Nathan Holbert, *holbert@tc.columbia.edu*
Department of Mathematics, Science, and Technology,
Teachers College, Columbia University, New York, USA

Michael Tissenbaum, *miketiss@illinois.edu*
Department of Curriculum and Instruction,
University of Illinois, Champaign, USA

Abstract

As more and more countries are pursuing the goal of integrating computing and computer science instruction into curricula and standards, it is important that we carefully consider what the goals and motivations of such programs are and whether or not it is in the best interest of the learners most directly impacted by them. While many national efforts tend to deploy rhetoric elevating economic concerns alongside statements about creativity and human flourishing, the programs, software, curricula, and infrastructure being designed and implemented focus heavily on providing learners with the skills, practices, and mindset of the professional software developer. We contend that computing for all efforts must take the "for all" seriously and recognize that preparing every learner for a career as a software developer is neither realistic nor desirable. Instead, those working towards the goal of universal computing education should begin to consider alternative endpoints for learners after completing computing curricula that better reflect the plurality of ways the computing is impacting their current lives and their futures. Further, we argue that constructionist designs and principles should play a central role in shaping what a computing education might look like that supports these diverse endpoints. In developing this argument, we provide examples of tools and environments that are designed towards alternative, yet equally valid and valuable, endpoints. Central to these alternative endpoints and the tools used to support learners are core constructionist ideas including the centrality of constructing computational artifacts, the importance of pursuing personally meaningful projects, and presenting learners with low-floor/high-ceiling tools with which to work.

Keywords

Constructionism, Computing Education, Computer Science Education, Equity, Access

Introduction

In recent years, there has been a concerted effort to make computing and coding a core educational experience in countries throughout the world (e.g. CSforAll, Make it Digital, Computing at School, etc.). A variety of arguments are given for these large scale efforts ranging from a desire to support young people in being able to "express themselves digitally" (BBC, 2019), empowering them to impact their communities through programming (Bhattacharya, 2017), to providing "the computational thinking skills they need to be creators in the digital economy" (Smith, 2016). These three goals of computing education--creative expression, social justice, and economic opportunity--are frequently cited as primary reasons all students should be exposed to the powerful ideas of computing. In their review of motivations for bringing computing instruction into all classrooms, Vogel and colleagues (2017) identified seven distinct motivations, adding arguments such as creating an informed citizenry and improving general technological literacy to the aforementioned goals. The diversity of these goals speaks to the way in which computing has become a core part of society. Furthermore, these goals highlight the need to make computing education efforts universal--targeting all young people regardless of school, age, or interest.

We think the goal of bringing computer science experiences and practices to all young people as part of their formal educational experience is a worthwhile endeavor and have each in our own way worked to support this effort. We are encouraged to see public rhetoric highlighting the social justice implications of computing education. Likewise, the explicit acknowledgment that computing is a powerful new form of creative expression aligns with a long history of computing education that emerged from early constructionist thought. And while we are hopeful that those participating in computing education are attending to these important goals, the enactment of these initiatives--in the form of curricula, learning environments, tools, assessments, policy, etc.--suggest that the most critical educational decision-makers most directly shaping the enactment of computing for all initiatives are prioritizing place economic concerns first. In other words, while the computing education community claims to attend to social justice and creative expression, the assumed endpoint of computing education seems to be about job preparation--increasing the number of programmers in the workforce so that we can compete in a global market.

Just as Papert explored the idea of alternative possible mathematics educations (1996), here we aim to identify just a few points in a large N-dimensional space that might serve as examples of possible versions of computing education. In this paper, we explore potential endpoints to computing education--what might people do with computing? We argue that computing education should truly acknowledge, and enact, the belief that computing has meaning and value in a host of potential careers and daily experiences. This recognition of alternative endpoints is important and potentially transformative. Identifying alternative potential computing educations invites us to define the dimensions in which this point exists—to, in essence, think about how legitimizing alternative endpoints beyond undergraduate degrees in computer science can bring new tools, practices, and contexts into computer science classrooms and change the narrative around what computer science is and what it looks like to practice it. As alternative endpoints become more central to computer science, characteristics of the curricula, tools, assessments, and projects that live in computer science classrooms can begin to change to reflect these alternative endpoints, and in doing so, can open up the field to those not historically drawn to conventional computer science pathways.

This paper argues that efforts to bring computing to all, and the learners who participate in such programs, would be better served by considering the plurality of endpoints beyond those that prioritize economic interests and career outcomes. Further, we argue that constructionist design and values should play a central role in shaping what a computing education might look like that supports these diverse endpoints. In doing so, various computing for all efforts can better welcome and support the learners they are trying to reach by aligning instruction and learning opportunities with the ideals, values, and goals of the learners. Further, legitimizing and valuing endpoints beyond conventional computer science careers can lead to a more inclusive and welcoming form of computer science, where creative, expressive, and culturally-valued instantiations of computer science ideas are valued alongside the skills that can lead to conventional computer science careers. This work adds to a growing chorus of voices, both in academia and beyond, pushing to rethink the goals, values, and priorities of contemporary computing education (Lewis, 2017; Santo et al., 2019; Vakil, 2018; Vogel et al., 2017).

To demonstrate this idea, this paper lays out three distinct computer science endpoints outside of the conventional computer science pipeline, showing how the consideration of alternative endpoints can shape the tools used, the ways learners are supported in engaging with computer science ideas, and ultimately reshape the computing education landscape and what it looks like for a learner to authentically participate in meaningful computing.

Motivation for Considering Alternative Computing Endpoints

This paper argues for a re-examination of the nature and goals of broad computing education initiatives. Instead of starting with specific values or goals, this work instead begins by considering various desired endpoints of computing instruction and then works backward to reason about what form learning activities might take and what are the underlying values and principles that support learners in reaching these endpoints. The result of this exercise is a push for rethinking the form of contemporary computing education with an eye towards more diverse, equitable, and meaningful endpoints.

Across the literature, a broad array of motivations are provided for computing education. Working with New York City school district stakeholders, Vogel et al. (2017) collected a total of 161 arguments for computer science instruction, and grouped them into seven categories: (1) economic and workforce development, (2) equity and social justice, (3) competencies and literacies, (4) citizenship and civic life, (5) scientific, technological and social innovation, (6) school improvement and reform and (7) fun, fulfillment and personal agency. This plurality of ideas is often not reflected in the nature of the tools, activities, and assessments used as part of classroom instruction. This is especially true with older learners where priorities further shift toward the use of professional programming languages and a prioritization for college and career readiness.

With this work, we introduce three distinct alternative endpoints for computing outside of the conventional computer science pathway as a means of rethinking what forms instruction can or should take.

Constructionism as a means to Reconceptualize Computing Education

A constructionist lens is a particularly powerful means for positioning alternative endpoints to computer education. Through the building of computational artifacts, learners have the opportunity to engage in critical reflection on what they are making and why and how it relates to them personally and to society more broadly (Ratto & Boler, 2014). By focusing on these broader socio-technical aspects of learners' construction (beyond the end-goal of getting a job as a programmer), we introduce opportunities for developing critical consciousness (Freire, 1974; Lee & Soep, 2016) and an understanding how computing shapes the world around them and their ability to create with it for their own goals, identity, construction, and expression (Holbert et al., in press; Tissenbaum et al., 2019).

Papert touted that children learn how to think critically through the process of solving problems that arise while programming computers (1996). Through the creative and investigative processes that are at the core of constructionism, learners begin to understand the multi-faceted ways that computing can and should be a central force for them to personally express themselves, construct their digital and personal identities, and empower them to be critically aware and empowered citizens.

Constructionism's attention to the learner's values and interests make it well suited to support learners in using computational power to explore a diverse range of experiences, practices, phenomena, etc. Whether supporting young people in constructing video games (Harel & Papert, 1991; Kafai, 1991; Weintrop et al., 2012), interactive art (Bontá et al., 2010; Papert & Solomon, 1971), musical instruments (Cavallo et al., 2004; Gorson et al., 2017), e-fashion (Buechley & Eisenberg, 2008; Kafai et al., 2014), or public service announcements (Blikstein, 2008), since the inception of the design paradigm, constructionists have cared deeply about supporting learners as they express their passions, explore their interests, or work to design solutions to real-world problems. We argue that this foundational quality must be present in any effort to broaden participation in computing education. When learners are given the space to construct objects--both digital or physical--that have personal or communal meaning, they have the opportunity to represent these passions in inspectable artifacts that can be viewed, critiqued, extended, or repurposed by others. This not only has powerful cognitive benefits--being able to externalize one's thinking into representational systems that can be debugged, modified, etc.--but also important identity implications. The computer code and resulting artifact can serve as a representation of one's work and one's contributions to the broader computing community. From a constructionist perspective, computing is not just an economically viable way to make things, but a way of *doing* things, with others, to change and impact the world.

Alternative Endpoints for Computer Science

In this section, we lay out three distinct views drawn from our research of alternative endpoints for computing education. The goal of this work is to argue for the importance of computing for all while also providing legitimate and authentic applications of computer science knowledge outside the existing pathway that leads to a computer science industry job.

Endpoint: Impacting local communities and immediate needs

The first endpoint we consider is the development of novices' identities as empowered to address real issues in their own lives, schools, and communities. While traditional computing education was locked to desktop computers, often taking place in computer lab settings, the introduction of mobile technologies (in particular smartphones) has allowed computing education to move out of the classroom and into learners' everyday lives. This ability for the products that students create to be taken out of the computer lab and into the world has allowed students and educators to move beyond simply writing code, instead critically asking *why* and *who* they are building it for, and to what end (Holbert, 2016; Lee & Soep, 2016). By situating computing education directly in students' lives, we open up computing education as a possibility space for impact and empowerment. This is critically important, as a long line of research has shown that the failure to meaningfully connect computing to the personal lives of students contributes to learners feeling computing is not useful or relevant to them (American Association of University Women, 1994; Couragion Corporation, 2018; Margolis & Fisher, 2003). This is particularly true for students underrepresented in computing and engineering careers (Cheryan et al., 2017; Pinkard et al., 2017; Taheri et al., 2019).

In response, we posit that there is a need to re-think the goals of computing education through a lens of *Computational Action* (Tissenbaum et al., 2019), which focuses on three key factors: 1) *Computational identity*, which is a person's recognition of themselves as capable of identifying and creatively implementing computational solutions to issues in their lives, schools, and communities; 2) *Digital empowerment*, which focuses on people's ability to put their computational identity into action in authentic and personally meaningful ways; and 3) *Computational design thinking*, in which learners' can successfully articulate the processes by which they will design and develop their solutions.

In order to support students' engagement in computational action, we need tools that reduce the barriers for them to quickly build, implement, and refine their designs. One example of the kinds of platforms particularly well-suited for such an approach is MIT's App Inventor, a block-based programming language that enables users to build fully functioning, native Android mobile applications. However, it is not enough to provide novice learners with a coding platform and simply let them loose. Supporting computational action also requires the development of scaffolds in the form of support materials (such as design documents) and scripted activities that lead students through the design process. Developing these additional supports is key to ensuring that students progress from ideation to implementation.

To explore how a computational action-focused curriculum can support students in developing meaningful solutions to personally-relevant issues, Tissenbaum, Sheldon & Ableson (2019) implemented a computational action curriculum in an ethnically diverse urban high school in the United States. Tissenbaum and colleagues chose this school as it encompassed a broad spectrum of students, particularly those not traditionally represented in the computing career pipeline. Working with the teacher, they identified an issue that was of interest to students at the school and the broader local community: the pollution of the local river (a major feature that runs through the middle of the city). Working in collaborative teams, students developed their own solutions to increase awareness and investigation strategies for cleaning up the river. To ensure that the students felt their work was meaningful (i.e. to support their *computational empowerment*), they presented their final projects at the school-wide job fair, which included visits from local council members and the mayor.

At the end of the curriculum, many of the students expressed that they never thought they would be able to build an app themselves, let alone build one that they felt had a chance to make real change. Many also expressed excitement towards developing solutions to new problems using the computational tools and knowledge developed during this project.

As this example shows, a computational action approach to computing education has the potential to support students to become, not only programmers but computationally literate, empowered problem-solving citizens.

Endpoint: Means of personal and social creative expression

Many computing initiatives and tools pursue the goal of empowering young people to express themselves digitally. These efforts see the computer and code as a digital canvas, a medium that enables a host of alternative forms of creative expression. Early implementations of Logo, the first true programming language for children, often invited young people to create "computer graphics" similar to those they saw at the arcade and on their video game systems (Harel & Papert, 1990; Kafai, 1996). Scratch, a successor of Logo and the programming environment most widely used to introduce learners to programming, invites children to create interactive stories, games, and animations (Resnick et al., 2009). Similarly, so-called "making," a popular means of combining computing with fabrication and craft work, invites learners to create personally interesting physical and tangible artifacts (Halverson & Sheridan, 2014).

While many computing education efforts do engage learners in personal and social creative expression, these activities are often used as a means to acquire STEM or computing content knowledge or practices. However, creative construction offers more than just a compelling way to encounter the practices of the software engineer. Here we propose that creative expression itself can be a powerful and worthwhile endpoint of computing education.

The design and creation of compelling artifacts that speak to the experiences, values, or perspectives of society has traditionally been considered the domain of the artist. While artists work with a variety of media and materials, the computer has been a useful tool for artists since its inception to ask questions about the nature of humanity, the role of technology in society, and to reflect on social and governmental structures and systems.

Afrofuturism is a genre of art, music, and literature that has used the practices, affordances, and implications of computing to great effect to imagine future societies and worlds that center the experiences and values of people of color (Anderson & Jones, 2015; Dery, 1994). These perspectives can be found in the costumes and pageantry of Parliament-Funkadelic, the stories sung by Janelle Monea's android alter ego Cindi Mayweather, and the futuristic technologies created by Shuri in the Black Panther. In the Remixing Wakanda project, Holbert and colleagues leveraged the Afrofuturist aesthetic and design genre to invite young people to reflect on the current state of the world and to use computational tools and practices to create artistic artifacts that construct a future that represents their values and perspectives (Dando et al., 2019; Holbert et al., in press).

In this project, making and computing became tools for Black teens to critically examine their experiences as young people of color in a large American city. Working with professional comic book artists, learning scientists, designers, and local activists, participants designed and ultimately constructed futuristic artifacts or societies that imagined futures that valued harmony between diverse groups of people and between humanity and nature. In these constructions, participants used computational tools, sensors, and circuitry to creatively merge aesthetic considerations with functionality to highlight humanity's problematic relationship with the environment and to acknowledge and respond to their experiences with racism and inequality (Figure 1). For example, one participant designed a fashionable cloak that hid the wearer from prying eyes--eyes that she said, "make you feel like you're alien [...] like you some art exhibit or something." While this cloak offered protection in the form of anonymity as well as a battery of sensors that monitored the health and wellbeing of the wearer (at one point the designer also considered including pepper spray as a built-in feature), the cloak also elevated a distinctly African aesthetic. Using textiles and patterns from her native Senegal, as well as a hypothetical technology that could morph into personally meaningful 3D iconography, this participant created a computing-rich artifact that proudly displayed her heritage.

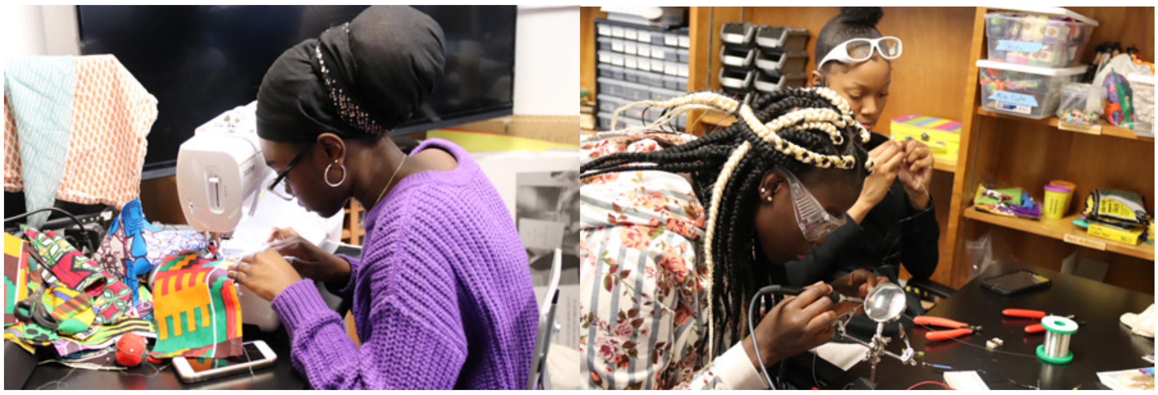


Figure 1. Learners constructing artifacts as part of the Remixing Wakanda project.

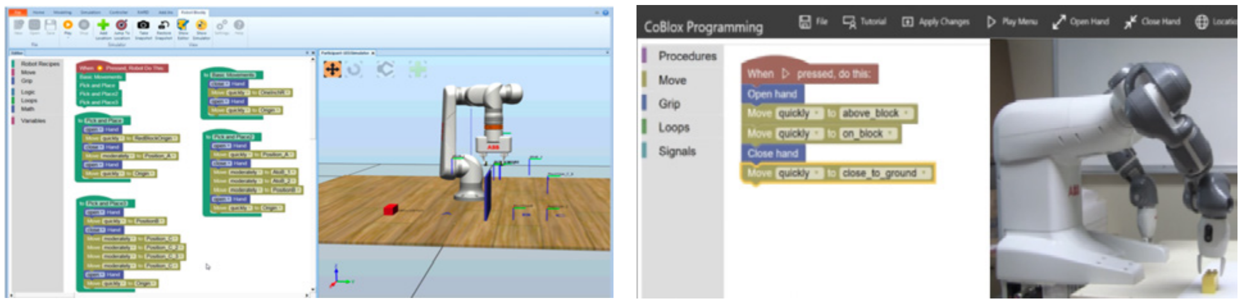
Another participant reflected on her personal frustration about litter and trash in the city. More than a visual blight, this trash often caused disruption in public transportation that impacted her ability to move through her city. As a response, this participant designed an aesthetically appealing trash receptacle that included a futuristic technology that would directly convert trash to energy that could be used to power street lights (a safety concern for those that work late) or serve as a charging station. She then went on to program a microcontroller to illustrate the principle behind this imagined technology.

In each case of the Remixing Wakanda project, computing serves as a tool for critically reflecting on the current state of the world and for creating representations of a possible future that might initiate change today. While participants did encounter coding, and potentially came away with new knowledge about computing concepts or practices, these experiences are themselves means towards the end of creative expression, of creating artifacts and representations the center their anxieties and fears as well as their hopes and dreams. While one implication of this work may be that the construction of critical artifacts may appeal to a broad range of learners currently underrepresented in computing domains, this is far from the only possible purpose of such design experiences. Rather, in the Remixing Wakanda project, computing is a tool for engaging in critical reflection on inequitable societies, unsustainable energy practices, and systems of oppression. Here, computing is a means of agitating for change--itself a powerful and important endpoint (Holbert, in press).

Endpoint: Blue Collar Computing

The third endpoint we present is the closest to the conventional endpoint of a profession in a computer science-related field but challenges the notion of what a computer science-related field looks like. As the technologies that enable automation become cheaper and their capabilities expand, the nature of manual labor is shifting. An example of this can be seen with collaborative robotics where humans and robots work side-by-side in a complementary capacity (Colgate et al., 1996; Kock et al., 2011). Where the robots excel at tasks that require precision or repetition, humans are more efficient at decision-making that requires judgment, adaptability, and creativity (Blank et al., 2006). Responding to this trend of the introduction of automation into new contexts, the skills and knowledge one needs to succeed in this setting relies on an understanding of foundational computing concepts in order to program and re-programming industrial robots. One way to prepare workers for the new computational landscape is to integrate computing across the K-12 landscape in an effort to prepare all learners to write and modify programs written in complicated industrial robotics programming languages. That approach aligns with much of the existing CS for all rhetoric which seeks to prepare all students for a future as a software developer. An alternative approach to address this issue is to redesign the tools at hand, in this case, the industrial robotics programming interface, so as to make it more intuitive, accessible, and draw more directly from the existing knowledge and expertise of today's worker.

Towards this end, Weintrop and colleagues set out to re-envision what it might look like to program industrial robots and investigated training approaches to help adult novices successfully author useful routines. The result of this work was a programming environment called CoBlox (Weintrop et al., 2018) which allows users with little or no prior programming to program virtual (Figure 2a) or physical (Figure 2b) robots. Drawing from prior work on the design of accessible and intuitive programming environments (Bau et al., 2017; Weintrop, 2019), CoBlox uses the block-based programming modality to situate the robotics programming task. Block-based programming using a programming-command-as-puzzle-piece metaphor to provide visual cues as to how and where a command can be used in the construction of a program (Maloney et al., 2010; Weintrop & Wilensky, 2015). CoBlox also leverages several features of block-based programming environments, such as natural language expressions within individual commands, predefined templates for common routines, and integration with both virtual and physical robots (Weintrop et al., 2017).



```
When ▷ pressed, do this:
Open hand
Move quickly to above_block
Move quickly to on_block
Close hand
Move quickly to close_to_ground
```

(a) (b)

Figure 2. Virtual (a) and physical (b) implementations of the CoBlox *programming Environment.*

The goal of highlighting the CoBlox design is to showcase what it looks like for computer science knowledge to be used in professional settings historically not considered within the purview of computer science. In documenting the ways that knowledge and practices clearly within the bounds of the discipline of computer science (programming in this case) can be enacted outside of what is typically viewed as a computer science endpoint, we show the importance of the consideration for alternative endpoints. As the skills and concepts from the field of computer science continue to impact a wider and more diverse set of professions, it is important for the narrative motivating and arguments justifying computer science to reflect this new plurality. With CoBlox, we see an authentic and legitimate professional endpoint in which computer science knowledge is valued but not typically included in the narrative around why computer science is important. By including endpoints that reflect the larger swath of professions impacted by computer science, students who do not see themselves as future software developers may come to recognize the utility of learning computer science.

Conclusion

The increasingly digital nature of our world requires that all learners feel empowered to understand and meaningfully participate in computational practices. The last decade has seen those from the computer science community lead the effort in designing the tools, creating curricula, and crafting the policy that will shape the form this instruction takes for future generations. While it is important for computer science to have a seat at the table, it is just as important that the ideas, values, and goals of those beyond the field also participate to reflect the growing role of computing in the world. Through envisioning and valuing alternative, yet equally valid and important, endpoints, this work seeks to start a conversation about the nature of the dimensions that might make up alternate computing educations--to re-evaluate the current tools and curricula to prepare learners for a future of active and empowered computing-literate citizens. In rethinking the goals of computing education, we see the ideas and principles of constructionism as having much to contribute towards realizing a form of computing education that more fully reflects the plurality and diversity of computing endpoints.

Motivating computing instruction solely based on economic outcomes does not accurately reflect the role of computing in the current and future society. In some contexts, the economic motivation for teaching computer science lives alongside goals such as preparing learners to be informed digital citizens or to prepare learners for 21st-century jobs beyond those in the technology sector. While such framing more accurately reflects the influence of computing in society, the programming languages, computer science curricula, and larger computing pathways still are largely designed as on-ramps intended to lead learners into the computing industry. In this way, the existing educational infrastructure is structured with computer science degrees and the industry jobs that will follow as the endpoint of computing instruction. While it is important that such pathways exist, it is equally important that instruction help learners recognize alternative endpoints computer science can lead to and support those learners for whom computer science can be a generative and valuable skill outside of professional contexts. By recognizing the value of these alternative endpoints, and hypothesizing diverse forms of computing educations, we open up the computing education landscape to be more inclusive, adaptive, and empowering for all, rather than for the select few who choose programming jobs as their educational endpoint.

References

American Association of University Women. (1994). *Shortchanging Girls, Shortchanging America*. AAUW Educational Foundation.

Anderson, R., & Jones, C. E. (2015). *Afrofuturism 2.0: The Rise of Astro-Blackness*. Lexington Books.

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: Blocks and beyond. *Communications of the ACM*, *60*(6), 72–80. https://doi.org/10.1145/3015455

BBC. (2019). *Make It Digital—The BBC micro:bit*. BBC. https://www.bbc.co.uk/programmes/articles/4hVG2Br1W1LKCmw8nSm9WnQ/the-bbc-micro-bit

Bhattacharya, A. (2017). *What happens when girls in one of the world's largest slums start coding and building apps*. Quartz India. https://qz.com/india/1032018/dharavi-diary-what-happens-when-girls-in-one-of-the-worlds-largest-slums-start-coding-and-building-apps/

Blank, D., Kumar, D., Meeden, L., & Yanco, H. (2006). The Pyro toolkit for AI and robotics. *AI Magazine*, *27*(1), 39.

Blikstein, P. (2008). Travels in Troy with Freire: Technology as an agent for emancipation. *Paulo Freire: The Possible Dream. Rotterdam, Netherlands: Sense*. http://tltl.stanford.edu/publications/papers-or-book-chapters/travels-troy-freire

Bontá, P., Papert, A., & Silverman, B. (2010). Turtle, Art, TurtleArt. *Proceedings of Constructionism 2010 Conference*.

Buechley, L., & Eisenberg, M. (2008). The LilyPad Arduino: Toward wearable engineering for everyone. *Pervasive Computing, IEEE*, *7*(2), 12–15.

Cavallo, D., Papert, S., & Stager, G. (2004). Climbing to understanding: Lessons from an experimental learning environment for adjudicated youth. *Proceedings of the 6th International Conference on Learning Sciences*, 113–120.

Cheryan, S., Ziegler, S. A., Montoya, A. K., & Jiang, L. (2017). Why are some STEM fields more gender balanced than others? *Psychological Bulletin*, *143*(1), 1–35. https://doi.org/10.1037/bul0000052

Colgate, J. E., Edward, J., Peshkin, M. A., & Wannasuphoprasit, W. (1996). *Cobots: Robots For Collaboration With Human Operators*.

Couragion Coroporation. (2018). *Altering the Vision of Who Can Succeed in Computing*. Oracle Academy.

Dando, M. B., Holbert, N., & Correa, I. (2019). Remixing Wakanda: Envisioning Critical Afrofuturist Design Pedagogies. *Proceedings of FabLearn 2019*, 156–159.

Dery, M. (1994). *Flame Wars: The Discourse of Cyberculture*. Duke University Press.

Freire, P. (1974). *Pedagogy of the Oppressed*. Basic Books.

Gorson, J., Patel, N., Beheshti, E., Magerko, B., & Horn, M. (2017). TunePad: Computational Thinking Through Sound Composition. *Proceedings of the 2017 Conference on Interaction Design and Children*, 484–489.

Halverson, E. R., & Sheridan, K. (2014). The maker movement in education. *Harvard Educational Review*, *84*(4), 495–504.

Harel, I., & Papert, S. (1990). Software design as a learning environment. *Interactive Learning Environments*, *1*(1), 1–32.

Harel, I., & Papert, S. (Eds.). (1991). *Constructionism.* Ablex Publishing.

Holbert, N. (In Press). Constructionism as a Pedagogy of Disrespect. In N Holbert, M. Berland, & Y. Kafai (Eds.), *Designing Constructionist Futures: The Art, Theory, and Practice of Learning Designs*. MIT Press.

Holbert, N, Dando, M. B., & Correa, I. (In Press). Afrofuturism as critical constructionist design: Building futures from the past and present. Journal of Learning, Media, and Technology. *Journal of Learning, Media, and Technology. http://dx.doi.org/10.1080/17439884.2020.1754237*

Holbert, Nathan. (2016). Leveraging cultural values and “ways of knowing” to increase diversity in maker activities. *International Journal of Child-Computer Interaction*, *9–10*, 33–39. https://doi.org/10.1016/j.ijcci.2016.10.002

Kafai, Y. (1996). Learning Design by Making Games: Children’s Development of Design Strategies in the Creation of a Complex Computational Artifact. In M. Resnick & Y. Kafai (Eds.), *Constructionism in Practice* (pp. 71–96). Lawrence Erlbaum. https://www.taylorfrancis.com/books/e/9780203053492/chapters/10.4324/9780203053492-11

Kafai, Y. B. (1991). Learning design by making games. In I. Harel & S. Papert (Eds.), *Constructionism* (pp. 71–96). Ablex Publishing Corporation.

Kafai, Y. B., Lee, E., Searle, K., Fields, D., Kaplan, E., & Lui, D. (2014). A Crafts-Oriented Approach to Computing in High School: Introducing Computational Concepts, Practices, and Perspectives with Electronic Textiles. *Trans. Comput. Educ.*, *14*(1), 1:1–1:20. https://doi.org/10.1145/2576874

Kock, S., Vittor, T., Matthias, B., Jerregard, H., Källman, M., Lundberg, I., Mellander, R., & Hedelind, M. (2011). Robot concept for scalable, flexible assembly automation: A technology study on a harmless dual-armed robot. *2011 IEEE International Symposium on Assembly and Manufacturing (ISAM)*, 1–5. https://doi.org/10.1109/ISAM.2011.5942358

Lee, C. H., & Soep, E. (2016). None But Ourselves Can Free Our Minds: Critical Computational Literacy as a Pedagogy of Resistance. *Equity & Excellence in Education*, *49*(4), 480–492. https://doi.org/10.1080/10665684.2016.1227157

Lewis, C. M. (2017). Good (and Bad) Reasons to Teach All Students Computer Science. In S. B. Fee, A. M. Holland-Minkley, & T. E. Lombardi (Eds.), *New Directions for Computing Education: Embedding Computing Across Disciplines* (pp. 15–34). Springer International Publishing. https://doi.org/10.1007/978-3-319-54226-3_2

Maloney, J. H., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), 16.

Margolis, J., & Fisher, A. (2003). *Unlocking the clubhouse: Women in computing*. The MIT Press.

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, *1*(1). https://doi.org/10.1007/BF00191473

Papert, Seymour, & Solomon, C. (1971). *Twenty things to do with a computer*. http://18.7.29.232/handle/1721.1/5836

Pinkard, N., Erete, S., Martin, C. K., & Royston, M. M. de. (2017). Digital Youth Divas: Exploring Narrative-Driven Curriculum to Spark Middle School Girls' Interest in Computational Activities. *Journal of the Learning Sciences*, *26*(3), 477–516. https://doi.org/10.1080/10508406.2017.1307199

Ratto, M., & Boler, M. (2014). *DIY citizenship: Critical making and social media*. MIT Press.

Resnick, M., Silverman, B., Kafai, Y., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., & Silver, J. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60.

Santo, R., Vogel, S., & Ching, D. (2019). *CS for What? Diverse Visions fo Computer Science Education in Practice*. CSforALL.

Smith, M. (2016). *Computer Science For All*. Whitehouse.Gov. https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all

Taheri, M., Ross, M., Hazari, Z., Weiss, W., Georgiopoulos, M., Christensen, K., Solis, T., Chari, D., & Taheri, Z. (2019). Exploring Computing Identity and Persistence Across Multiple Groups Us-ing Structural Equation Modeling. *American Society for Engineering Education (ASEE) Conference Proceedings*.

Tissenbaum, M., Sheldon, J., & Abelson, H. (2019). From computational thinking to computational action. *Communications of the ACM*, *62*(3), 34–36.

Vakil, S. (2018). Ethics, Identity, and Political Vision: Toward a Justice-Centered Approach to Equity in Computer Science Education. *Harvard Educational Review*, *88*(1), 26–52.

Vogel, S., Santo, R., & Ching, D. (2017). Visions of Computer Science Education: Unpacking Arguments for and Projected Impacts of CS4All Initiatives. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education - SIGCSE '17*, 609–614.

Weintrop, D. (2019). Block-based Programming in Computer Science Education. *Commun. ACM*, *62*(8), 22–25. https://doi.org/10.1145/3341221

Weintrop, D., Afzal, A., Salac, J., Francis, P., Li, B., Shepherd, D. C., & Franklin, D. (2018). Evaluating CoBlox: A Comparative Study of Robotics Programming Environments for Adult Novices. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 366:1-12. https://doi.org/10.1145/3173574.3173940

Weintrop, D., Holbert, N., Wilensky, U., & Horn, M. S. (2012). Redefining constructionist video games: Marrying constructionism and video game design. In C. Kynigos, J. Clayson, & N. Yiannoutsou (Eds.), *Proceedings of the Constructionism 2012 Conference*.

Weintrop, D., Shepherd, D. C., Francis, P., & Franklin, D. (2017). Blockly goes to work: Block-based programming for industrial robots. *2017 IEEE Blocks and Beyond Workshop*, 29–36. https://doi.org/10.1109/BLOCKS.2017.8120406

Weintrop, D., & Wilensky, U. (2015). To Block or Not to Block, That is the Question: Students' Perceptions of Blocks-based Programming. *Proceedings of the 14th International Conference on Interaction Design and Children*, 199–208. https://doi.org/10.1145/2771839.2771860