Investigating Creative and Critical Engagement with Computing in the *Hour of Code* (Practical Report)

Luis Morales-Navarro
luismn@upenn.edu
University of Pennsylvania
Philadelphia, PA, USA

Yasmin B. Kafai
kafai@upenn.edu
University of Pennsylvania
Philadelphia, PA, USA

Gayithri Jayathirtha
gayithri@upenn.edu
University of Pennsylvania
Philadelphia, PA, USA

Mia Shaw
mshaw12@upenn.edu
University of Pennsylvania
Philadelphia, PA, USA

ABSTRACT

The Hour of Code provides brief stand-alone activities to introduce K-12 learners to computing concepts and applications. While these activities have successfully reached hundreds of millions of students around the globe, there are calls for more creative and critical engagement with computing. In this paper, we examine the creative and critical content of 316 Hour of Code activities offered to middle school and high school students as part of the official 2020 Computer Science Education Week. Our content analysis revealed that only 13% Hour of Code activities promoted creative engagement while only 1% of Hour of Code activities focused on critical engagement, offering discussion guides at best. In the discussion we provide recommendations for designing for more critical and creative engagement with computing in future Hour of Code or similar hour-long activities.

CCS CONCEPTS

• **Social and professional topics → K-12 education**; **Computing literacy**.

KEYWORDS

Critical computing, Hour of Code, Scratch, Programming, Creative computing

ACM Reference Format:
Luis Morales-Navarro, Yasmin B. Kafai, Gayithri Jayathirtha, and Mia Shaw. 2021. Investigating Creative and Critical Engagement with Computing in the *Hour of Code* (Practical Report). In *The 16th Workshop in Primary and Secondary Computing Education (WiPSCE '21), October 18–20, 2021, Virtual Event, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3481312.3481314

1 INTRODUCTION

The Hour of Code (hereafter: HoC) is an annual event that offers hour-long activities to introduce K-12 youth to computing [35], taking a first step in "unlocking" the proverbial closed doors to the clubhouses of computing [21]. According to HoC's organizers, since its launch in 2013, over 1,123,794,533 youth have participated in the annual event. In December 2020 alone, 68,000+ HoC events were offered in 180 countries around the globe amidst the many efforts to introduce and promote computing and coding among youth. By any measure, the outreach HoC achieved over the last eight years has been an unprecedented success [13] but also not without critics. Efforts such as the HoC overemphasize the importance of training people for programming jobs over the potential of code as a means to promote youth's expressive, aesthetic, rhetorical, and critical abilities [32–34].

The purpose of this paper is to (1) examine how 2020 HoC activities promote creative and critical computing and (2) to articulate design directives for developing HoC-type activities that can expand the introduction to computing by promoting reflective critical engagement with the field. Our analysis focuses on activities offered to middle and high school youth (grades 6+) during the 2020 HoC to address the following research question: How did HoC 2020 activities promote creative and critical engagement with computing? We discuss possible directions for introducing and expanding critical and creative engagement with computing in HoC-type activities.

2 BACKGROUND

HoC was launched in 2013 in the United States by the Computer Science Teacher Association (CSTA) and *code.org*, a tech-industry-funded non-profit organization, to introduce youth to computing and advocate for making computing a part of the standard national curriculum [2]. However, the use of brief stand-alone activities to introduce K-12 youth to computing started much earlier. Scratch Day, an annual event, could be considered a precursor to HoC. The first Scratch Day in 2008 (following the 2007 launch of the Scratch programming language and community) was organized by MIT, the university that also hosts the Scratch website. Today, a little over a decade, these events are community-organized and take place in local schools, clubhouses, and universities around the globe. Scratch Day meeting locations and times are published on a website (https://day.scratch.mit.edu) thus allowing interested youth and their parents to find events close to home. Scratch Days can last a full afternoon, day or even a weekend, offering multiple introductory activities to Scratch and other tools [22].

In contrast, from the beginning, the HoC was designed to provide hour-long activities that teachers could use in their classes during Computer Science Education week (CSEdWeek), scheduled every year in December. Over the years, HoC has spread internationally, being adopted by countries across all continents (e.g., [3, 8, 10]). Considering the number of participants around the globe, HoC has been successful in terms of broadening access and giving K-12 students a first hand experience with computing [13]. Yet, only few studies have taken a closer look at what students do and gain from HoC events, moving beyond the numbers of participants and lines of code to examine HoC's possible impact in promoting interest in coding [24], learner motivation [23] or opening pathways to STEM careers [1]. At the outset, much of the early criticism of HoC has focused on the length and type of activities being limited to involving individual students in "puzzle-like" programming activities [27].

Recent criticism of introductory computing activities, such as HoC, has noted that very often "students do not have the opportunity to experience the full conceptual and expressive powers of coding" [25] (p. 121), and that computing is presented as a "value-neutral tool independent from society" [19] (p. 31) without considering its societal and ethical implications [30, 31]. As such, it is crucial to investigate how creative and critical computing is promoted in introductory activities.

Advocates for creative computing frame computing as a tool for design and self-expression where learners build on personal interests and connections to their lives [6, 18]. Creative computing provides opportunities for learners to engage with concrete experiences and create computational artifacts that are expressive rather than utilitarian [20]. Indeed, this approach to computing emphasizes that learning programming can be an opportunity to express, share and develop creativity rather than just technical skills [5, 7]. Over the years, HoC has attempted to provide creative opportunities by offering a wider range of activities in diverse programming tools and platforms (e.g., Scratch, Python, Minecraft)[12]. Yet, although the catalogue of activities is now broader and more diverse, with some activities focusing on creative design and expression, most efforts have centered on the inclusion of popular commercial content in puzzle-like activities (e.g., Cartoon Network and Disney characters, Google Doodles).

Proponents of critical computing take into account socio-cultural and political contexts and advocate for attending to the implications, consequences, and limitations of computing [19, 30]. Here criticism of HoC and similar activities has brought attention to the hidden curriculum that promotes limited perspectives about who can and should code and even why everyone should learn to code [33, 34]. Furthermore, there is a need to counterbalance overly technocentric views of computing applications as solutions to structural inequities that shield how computing itself can benefit some people while harming others based on race, gender, and class [4, 9]. These criticisms provide pointers to how HoC activities can be shaped to address critical issues.

In this paper, we want to examine how far along HoC in 2020 has come in terms of addressing the creative and critical dimensions of computing in its activities. Given HoC's widespread reach and worldwide recognition as *the* introduction to computing for most K-12 students, it is imperative to understand the messaging about computing promoted in these activities. To do so, we draw on previous work that has articulated design principles for promoting creative computing via construction kits [26] and computing activities [17] and developed a new one to address critical dimensions that can be applied to HoC activities. To design, and by extension evaluate, computing activities we should address: (1) *Low Floors*—the activity should be intuitive enough for new users to acclimate to it gradually and with a degree of confidence; (2) *High Ceilings*—the activity should also facilitate more experienced users to create increasingly complex applications that grow more intricate and nuanced as their proficiency using the tool increases; and (3) *Wide Walls*—the activity should allow for a wide range of projects, letting users tap into their personal experiences as well as popular culture to design and develop projects entirely unique and representative of their own interests and backgrounds. In addition, we consider a fourth principle (4) *Open Windows*—the activity should facilitate participation and sharing computational media creations with the broader community [16].

Current HoC activities address mostly the first principle, *Low Floors*, by lowering the threshold for beginners to start learning to code. Little attention has been paid to principles such as *High Ceilings*, given the introductory and short term nature of HoC. Over the years, HoC has tried to address the *Wide Walls* principle by including a broader and more diverse set of activities and programming environments, with some focusing on creative designs. Activities that foster *Open Windows* are much less present because of the individual puzzle-like nature of HoC activities. Whenever there is sharing and discussion these are limited to the classroom and usually directed by teachers.

Explicit criteria for evaluating critical computing have not been developed so far. The principles of *low floors*, *wide walls*, *high ceilings*, and *open windows* should not only apply to creative computing but also to activities in which youth engage with critical computing, as these ensure that activities are accessible and of interest to youth. To continue the metaphor of the house and address critical issues within computing, we propose a fifth principle, which we call (5) *Visible Foundations*—the tool or activity should also raise questions and allow novices to engage critically with certain foundational ideas about what computing is, who can participate in computing and what the purposes and implications of computing are.

In 2019, CSTA and *code.org*, in response to rising concerns about the lack of diversity in the tech industry and hidden biases in computing, took a first step towards critical computing and created HoC activities under the umbrella of CS4Good to address environmental, gender, and LGBTQ+ issues with (but not within!) computing. In examining the *visible foundations* of HoC activities, we want to pay attention to how critical issues within computing can be made more transparent. In this paper, we analyze the *wide walls* and *visible foundations* of HoC 2020 activities to understand where and how these activities stand in promoting creative and critical engagement with computing.

3 METHODS

3.1 Hour of Code Activities

We examined learning activities listed on *code.org*'s *hourofcode.com* and CSTA's *csedweek.org*. Activities on the HoC website are mainly contributed by *code.org* partners (technology companies and educational organizations). *code.org* invited and screened submissions of HoC activities in October 2020. Accepted submissions were listed on the site and often included short promotional videos to provide an overview, in addition to an outline of the activity. Activities on the platform could be filtered by age, topic, and programming language. In 2020 over 416 beginner activities were listed, with 264 for elementary, 319 for middle school and 235 for high school; some activities were listed for multiple age groups. Additionally, in 2020, CSTA's CSEdWeek website featured 11 CS4SocialJustice Heroes with activity cards and 13 CS4SocialJustice activities.

1.png

Figure 1: Sampling of HoC projects on *code.org* website.

3.2 Data Collection and Analysis

We focused our data collection and analysis on the publicly available beginner HoC activities offered to middle and high school students and teachers on *code.org*'s *hourofcode.com* and CSTA's *csedweek.com*. To better understand the kind of activities that were offered in the 2020 Hour of Code, we systematically selected a set of HoC activities and conducted content analysis on them. To scope our analysis, using the filtering feature of the HoC website, we selected 1-hour long activities offered to beginner middle and high school (grades 6-8 and grades 9+) students and teachers. All activities with working hyperlinks to activity details were included in our analysis. Listed activities which did not have any working hyperlinks to the activity details at the time of the analysis were not included. Following, we analyzed the content of 316 learning activities—303 available 1-hour long *code.org* HoC beginner activities for middle school and high school and 13 activities available on CSTA's CSEdWeek site. Two authors independently conducted inductive descriptive coding [28] of all the HoC activities to capture different qualitative aspects of these activities and their promotion of creative and critical engagement with computing. In our first round of analysis we created codes to describe the activities according to their type of engagement in terms of *wide walls*, CS4Good/CS4SocialJustice themes, use of media, programming languages and environments, and the presence of commercial or corporate content. In a second round of analysis we focused on *visible foundations* and analyzed the projects with CS4Good/CS4SocialJustice themes and developed a coding scheme to analyze how these activities engage or not with critical computing. We applied the scheme across activities and discussed ambiguity and differences between coders until we reached consensus.

4 FINDINGS

Overall, HoC's website and CSTA's CSEdWeek website offered over 316 activities for youth in middle school and high school that included games, cartoons, and various other activities (see Figure 1); in addition, in 2020 several activities also addressed the COVID-19 crisis. . HoC activities introduced computing using a variety of programming languages and environments (see Figure 2). For example, 47.5% of activities used blocks (e.g., Scratch, AppInventor, Code.org Blocks), 17.5% used web programming and markup languages (JavaScript, HTML, CSS), 12% used Python, and 6% used other programming languages such as C++ or Java. Some activities (13.5%) included unplugged programming, using paper cards, worksheets or human bodies to explore programming concepts, and few activities (4%) had no programming at all.

4.1 *Wide Walls* for Creative Computing

In terms of *Wide Walls*, although HoC offered a wide range of activities, with diverse contexts, characters, topics, and programming languages and environments, notably only 13% of the activities guided learners to create projects that tap into their personal interests (see Figure 2). The large majority of HoC activities (85%) were close-ended, with students following step-by-step tutorials. For example, a Disney sponsored activity, "Moana: Wayfinding with Code" guided learners with step-by-step instructions to use programming blocks to make the character move around and complete tasks. Any attempt of creative exploration (trying different things, playing around with the characters, or coming up with new paths and solutions) was discouraged by flagging error messages. Some activities (12.5%) provided guided open-ended opportunities

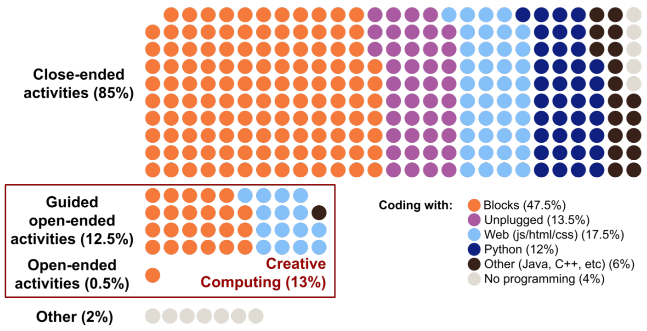

Figure 2: Visualization of HoC activities (n=316) by type activity offered to students and teachers. Color indicates the programming language used in the activity.

with scaffolding for students to create projects related to their own interests (see Figure 3). "Imagine a World," for example, invited students to imagine worlds where anything is possible, "where animals could talk or people could fly." In this activity learners were guided through the different programming blocks and features they could use to make creative Scratch projects with themes, characters, sounds and actions of their own choice. Another activity, "Create a typeface," introduced students to programming by showcasing basic drawing functions in a JavaScript library and later inviting learners to use these functions to design their own typefaces. In both these activities, students were scaffolded to create personalized projects that use and apply specific programming concepts. Only one activity, Brain Pop's "Engineering and Tech Creative Coding," was open-ended, allowing students to come up with their own project ideas without direct guidance. This activity included a short video that introduced learners to the programming interface and although it offered several suggestions of types of projects students could make, it left it up to learners to explore the tool and come up with their own project ideas.

DESIGN YOUR OWN TYPEFACE
ENGINEERING & TECH: COMPUTER PROGRAMMING (SCRATCH)
Imagine a World
Grades 2+ | Blocks
Create your own typeface!
Grades 6+ | JavaScript
Engineering & Tech: Computer Progra...
Grades 6+ | Blocks, JavaScript

Figure 3: Examples of Creative Computing HoC activities on *hourofcode.com*.

4.2 *Visible Foundations* for Critical Computing

Of the 316 analyzed activities, 46 (13.29%) included CS4Good and CS4SocialJustice topics (see Figure 5) that engaged with environmental (13 activities), gender (6 activities), health and wellness (8 activities), and race issues (5 activities) among others. We identified three ways in which these 46 HoC activities addressed CS4Good and CS4SocialJustice topics: 24 activities used coding to raise awareness about issues outside of computing, 16 activities used coding to create applications that fix issues in the world, and only 4 activities somewhat made *visible* the otherwise invisible *foundations* by addressing critical issues within computing as a field. For example, in the "Using CS to Raise Awareness" category, an activity called "Beaver in a Mask" guided students to create an app that gives points to users for identifying the correct use of facemasks to mitigate the spread of COVID-19. Other activities in this category guided students to create public service announcement animations

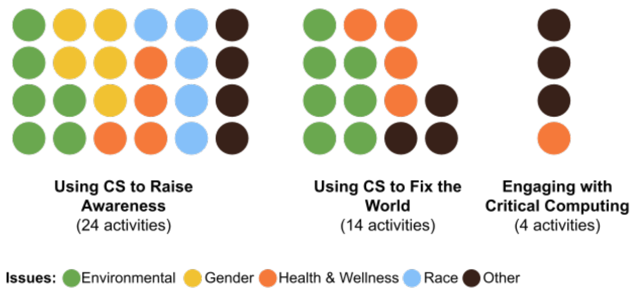

Figure 4: Visualization of HoC activities with CS4Good and CS4SocialJustice topics by category.

that addressed how to prevent COVID-19, plastic pollution, and marine wildlife degradation. Activities that focused on "Using CS to Fix the World" included creating meditation apps, a healthy meal app, and training an AI algorithm to identify plastic in the ocean. None of the activities in these two categories acknowledged the limitations of such solutions or questioned the possible implications of using code and technology to raise awareness or "fix" issues, reinforcing the idea of technological solutionism [29], that is the naïve belief that technology will solve humanity's greatest ills.

In the "Engaging with Critical Computing" category, we identified four activities that addressed *Visible Foundations*. Yet, these mostly focused on talking about computing, rather than having students code. As such, it is not a coincidence that all four activities require strong teacher guidance and facilitation. CSTA's "CS4SocialJustice Heroes," for example, included a robust site with posters, videos, and discussion questions around people who do critical work in computing such as Nicki Washington, Ruha Benjamin, and Joy Buolamwini. Another activity, called "Queering Computer Science," also on CSTA's portal, provided a guide for teachers to address critical issues about gender identity and sexual orientation in computer science and at their intersection with race and disability. The only activity from *code.org*'s HoC portal in this category, called "AI & Drawing," provided teachers with a guide to address algorithmic bias in machine learning by having conversations around Google's "QuickDraw" game.

In contrast, "Critical Computational Thinking: West Virginia – Appalachia as Context for Learning," on CSTA's site, provided teachers with a guide to discuss algorithmic bias and have students code a food access and distribution application to address food deserts. The guide invited teachers and students to interrogate the values embedded in their applications and how, while aiming to solve a social issue, software developers can generate further inequities; all of this with the goal of coming up with plausible ways to design applications that respond to the real needs of their community. Overall, despite the large number of activities available as a part of the HoC, very few engaged students in creative ways and with critical aspects of representation, implications, participation, and ethics within computing.

AI & Drawing
Grades 6+ | Language independent (can be taught in multiple languages)
WHO THRIVES? WHO DIES?
WHOSE BODY MATTERS?
WHOSE HISTORY SURVIVES?
WHO GETS POLICED?
WHO IS SAFE?
WHO IS MISSING?
Queering Computer Science
This was part of a short presentation given at the CSTA New England Regional Conference "Equitable Remote Learning."
Grades K - 12
Critical Computational Thinking
This example activity focuses on how computational thinking can be used as a critical lens to help students tackle a particular social justice issue.
Grades 3 - 8

Figure 5: Examples of critical computing activities on *hourofcode.com* (left) and *csedweek.org*(center and right).

5 DISCUSSION

In this paper, we examined the public portfolio of 2020 HoC activities. While HoC 2020 offered a large number of introductory activities (over 400 for all K-12 groups) in a variety of contexts and programming environments, only a small subsection promoted creative and critical engagement in the introduction of computing. In the following sections, we discuss what we see are challenges to creative and critical HoC activities, how we can design creative and critical HoC activities, and how we can prepare teachers to facilitate these activities.

5.1 Engagement with Creative and Critical Computing in HoC 2020

In our analysis of 2020 HoC activities we found that few activities (13%) engage students in creative computing by making projects that relate to their personal interests and even fewer (1%) offered opportunities to engage with computing critically. This might be due to the puzzle-like traditional style of HoC activities [27]. However, it is promising that 12.5% of HoC activities were guided open-ended activities. These activities scaffolded creative computing and engaged with the *Wide Walls* principle by guiding learners through the introduction of programming concepts to create projects that were connected to their personal interests. It is not a coincidence that only one activity was completely open-ended, given the time constraint of HoC activities. Perhaps what works best is to provide guided open-ended activities where students can be guided as they are introduced to programming but also have the space to create personally meaningful artifacts.

Most CS4SocialJustice and CS4Good activities reproduced the ideas that CS can help raise awareness about social issues or "fix" the world and very few engaged with the *Visible Foundations* critical computing principle. The few HoC activities that focused on critical engagement offered discussion guides at best. These activities made room for discussions about critical issues in computing instead of providing coding opportunities where critical issues were considered. Only one activity, "Critical Computational Thinking: West Virginia – Appalachia as Context for Learning," provided opportunities to both code and discuss the implications of the applications students were making. From this activity we can learn that scaffolding is important, that discussing the implications of computing works better when these are relevant to the communities of the students and to the projects/applications they are coding. Within the time constraints of HoC, we need to understand how to better support students to both engage in critical conversations and adopt some perspectives with regards to their application designs. Activities can be set in a context of investigating issues of representation, participation, or implications within computing to encourage learners to explore design dimensions such as whom they are designing for, what issues they are addressing, who can and cannot use their applications, or what are the implications of their projects for communities. Such questions can further integrate critical aspects with programming projects, instead of treating them in isolation.

5.2 Moving Forward: Designing Creative and Critical HoC Activities

This analysis provides important insights for how we can design future HoC introductory activities, with *Wide Walls* and *Visible Foundations*, that address creative and critical computing. First, we obviously want to keep the central premise of *Low Floors* for HoC activities which is that they should be accessible. This principle should also apply to any activities that engage youth critically with computing. Second, we need to work on *Widening Walls* by providing multiple ways of addressing critical questions within computing in HoC activities: who is coding, for whom are they coding, towards what ends, and what are the implications of computing applications. We can design an array of activities exploring a question at a time to allow students to delve deeper into these critical issues, approaching different angles and perspectives (designer, user, programmer, etc.). Having a multi-faceted approach to critical issues may help engage a broader set of students in these conversations, attending to different concerns students may be interested in.

We found HoC examples that illustrated compelling ways to engage K-12 youth with imagining future worlds. For instance, in one HoC Scratch activity, discussed above, youth were asked to imagine future worlds in which "animals could talk or people could fly." This same activity could be leveraged to imagine future worlds that address critical issues. For example, in their work on Remixing Wakanda, Holbert, Dando and Correa [14] suggest that critical constructionist design could invite "youth to critically examine social, economic, and environmental systems by both connecting to personal and family histories as well as reflecting on local and lived experiences … [to] design futuristic artifacts that critique existing social inequities and environmental instability." (p. 1). In computing education, plenty of attention has been given to how learning computational concepts like variables, conditionals and loops can be scaffolded. Now, we need to learn how to scaffold conversations about how computing applications have implications on people, communities, and the environment. While discussing these issues is one way to address them, it is only the beginning. We need to further support students to integrate these ideas into the computing applications they design.

Third, we must work on *Opening Windows* by providing audiences for sharing and discussing critical issues in computing. Right now, teachers use HoC activities with their individual classes. It is likely that within those classes students share and compare artifacts produced during HoC. Yet, HoC can go beyond, we can *Open Windows* by having youth share their creations with other HoC participants across the globe. For example, activities that use Scratch can create studios where students can share their projects (e.g., [15]). Even further, *Open windows* can also involve bringing in multiple perspectives and students' lived experiences to how we introduce computing and programming in HoC, this could be by connecting to cultural traditions, drawing inspiration from other student HoC projects or designing HoC activities that require student collaboration inside the classroom and with students' wider communities.

5.3 Preparing Teachers for Critical Discussions

In order to introduce students to critical computing in HoC-style activities we have to prepare teachers to facilitate learning about computing concepts while thinking about computing critically. This is particularly important considering that many teachers that facilitate HoC activities are not necessarily CS teachers. Even if facilitators are CS teachers, they might not have previous experience facilitating conversations about computing and society. For instance, an activity that aims to support students to learn about conditional statements and their role in shaping decision-making algorithms in job markets and policing systems that are biased against women and people of color would require teachers to integrate disciplines with epistemological differences and bring together computational concepts in dialogue with their societal implications. While the computing education field has ample evidence, accounts, and support for the former, we barely know how to support teachers with the latter. To start with, we need to support teachers in facilitating critical conversations, by providing supplementary materials, while also supporting learners without overwhelming them. Many HoC 2020 activities were accompanied by short video trailers that introduced the activities and provided overviews of tools used. These trailers can give examples of completed projects and even testimonials, making the activities more accessible to newcomers. Many teachers might also use these trailers to decide what HoC activities to choose for their classes. But more importantly, short videos have the potential of becoming conversation starters for critical issues in computing. For this, we need to generate more scaffolds and materials to help teachers prepare and steward critical discussions.

6 CONCLUSION

While HoC activities are short in nature, they nonetheless have reached hundreds of millions of youth around the globe. The range of HoC activities offered, thus, signals to youth what is considered representative of computing, what is valued within the field and in which ways computing can address but also raise issues. We argue that engaging with computing creatively and critically has to start early, coupled with the introduction of computing concepts. This is not only to justify learning to code but also to prepare youth to create and express themselves with code and understand and confront the challenges associated with computing technologies. Following educator and activist Paulo Freire's saying that "reading the word is not only preceded by reading the world, but also by a certain form of writing it or rewriting" [11] (p. 18), we see that reading and writing code is as much about reading and writing the world as it is about understanding, changing, and re-making the world in which we live—and this involves thinking creatively and engaging with critical issues.

REFERENCES

[1] Ugochi Acholonu, Jessa Dickinson, Dominic Amato, and Nichole Pinkard. 2016. Lessons learned from hosting an Hour of Code event. In *2016 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*. IEEE, IEEE, 1–4.

[2] William Aspray. 2016. *Participation in Computing: The National Science Foundation's Expansionary Programs* (1st ed.). Springer Publishing Company, Incorporated.

[3] Ashok Basawapatna, Alexander Repenning, Mark Savignano, Josiane Manera, Nora Escherle, and Lorenzo Repenning. 2018. Is drawing video game characters in an hour of code activity a waste of time?. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 93–98.

[4] Ruha Benjamin. 2019. *Race After Technology: Abolitionist Tools for the New Jim Code*. Wiley. https://books.google.com/books?id=G6-hDwAAQBAJ

[5] Matthew Berland, Nathan Holbert, and Yasmin B. Kafai. 2020. Introduction: Fifty Years Of Constructionism. In *Designing Constructionist Futures The Art, Theory, and Practice of Learning Designs*, Matthew Berland, Nathan Holbert, and Yasmin B. Kafai (Eds.). MIT Press, Cambridge, MA, 1–20.

[6] Karen Brennan, Christian Balch, and Michelle Chung. 2014. Creative computing. *Harvard Graduate School of Education* (2014).

[7] Karen Brennan and Jimenez Raquel. 2020. The Scratch Educator Meetup: Useful Learning In A Playful Space. In *Designing Constructionist Futures The Art, Theory, and Practice of Learning Designs*, Matthew Berland, Nathan Holbert, and Yasmin B. Kafai (Eds.). MIT Press, Cambridge, MA, 1–20.

[8] Carlos Jorge Brigas and José Alberto Quitério Figueiredo. 2019. " The Hour of the Code": Computational Thinking Workshop in a Primary School in Guarda, Portugal. *Research in Social Sciences and Technology* 4, 2 (2019), 129–136.

[9] Sasha Costanza-Chock. 2020. *Design justice: Community-led practices to build the worlds we need.* MIT Press.

[10] Nora Escherle, Dorit Assaf, Ashok Basawapatna, Carmine Maiello, and Alexander Repenning. 2015. Launching swiss computer science education week. In *Proceedings of the Workshop in Primary and Secondary Computing Education*. 11–16.

[11] Paulo Freire. 1985. Reading the world and reading the word: An interview with Paulo Freire. *Language arts* 62, 1 (1985), 15–21.

[12] Reza Ghasem Aghaei, Ali Arya, and Robert Biddle. 2017. Affective walkthroughs and heuristics: Evaluating minecraft hour of code. In *International Conference on Learning and Collaboration Technologies*. Springer, Springer, 22–40.

[13] Michael Halvorson. 2020. *Code Nation: Personal Computing and the Learn to Program Movement in America.* Association for Computing Machinery.

[14] Nathan Holbert, Michael Dando, and Isabel Correa. 2020. Afrofuturism as critical constructionist design: building futures from the past and present. *Learning, Media and Technology* 45, 4 (2020), 328–344.

[15] Yasmin Kafai, Gayithri Jayathirtha, Mia Shaw, and Luis Morales-Navarro. 2021. CodeQuilt: Designing an Hour of Code Activity for Creative and Critical Engagement with Computing. In *Interaction Design and Children*. 573–576.

[16] Yasmin B Kafai and Quinn Burke. 2014. *Connected code: Why children need to learn programming.* MIT Press.

[17] Yasmin B Kafai and Deborah A Fields. 2018. Some reflections on designing constructionist activities for classrooms. *Proceedings from Constructionism* (2018).

[18] Yasmin B Kafai and Kylie A Peppler. 2009. Creative coding: Programming for personal expression. In *The 8th international conference on computer supported collaborative learning*. 76–78.

[19] Amy J Ko, Alannah Oleson, Neil Ryan, Yim Register, Benjamin Xie, Mina Tari, Matthew Davidson, Stefania Druga, and Dastyni Loksa. 2020. It is time for more critical CS education. *Commun. ACM* 63, 11 (2020), 31–33.

[20] Golan Levin and Tega Brain. 2021. *Code as Creative Medium: A Handbook for Computational Art and Design.* MIT Press.

[21] Jane Margolis and Allan Fisher. 2002. *Unlocking the clubhouse: Women in computing.* MIT press.

[22] Pedro Plaza, Elio Sancristobal, German Carro, Manuel Castro, and Manuel Blazquez. 2018. Scratch day to introduce robotics. In *2018 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, IEEE, 208–216.

[23] Alexander Repenning and Ashok Basawapatna. 2016. Drops and Kinks: Modeling the Retention of Flow for Hour of Code Style Tutorials. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education*. ACM, 76–79.

[24] Alexander Repenning, Ashok Basawapatna, Dorit Assaf, Carmine Maiello, and Nora Escherle. 2016. Retention of flow: Evaluating a computer science education week activity. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 633–638.

[25] Mitchel Resnick and Natalie Rusk. 2020. Coding at a crossroads. *Commun. ACM* 63, 11 (2020), 120–127.

[26] Mitchel Resnick and Brian Silverman. 2005. Some reflections on designing construction kits for kids. In *Proceedings of the 2005 conference on Interaction design and children*. 117–122.

[27] Margarida Romero, Ann-Louise Davidson, Giuliana Cucinelli, Hubert Ouellet, and Kate Arthur. 2016. Learning to code: from procedural puzzle-based games to creative programming. *Revista del Congrés Internacional de Docència Universitària i Innovació (CIDUI)* 3 (2016).

[28] Johnny Saldaña. 2013. *The coding manual for qualitative researchers* (2nd ed.). Sage.

[29] Kentaro Toyama. 2015. *Geek heresy: Rescuing social change from the cult of technology.* PublicAffairs.

[30] Sepehr Vakil. 2018. Ethics, identity, and political vision: Toward a justice-centered approach to equity in computer science education. *Harvard Educational Review* 88, 1 (2018), 26–52.

[31] Sepehr Vakil. 2020. "I've Always Been Scared That Someday I'm Going to Sell Out": Exploring the relationship between Political Identity and Learning in Computer Science Education. *Cognition and Instruction* 38, 2 (2020), 87–115.

[32] Annette Vee. 2017. *Coding literacy: How computer programming is changing writing.* MIT Press.

[33] Sara Vogel. 2019. Power, Discourse, and Knowledge in Computer Science Education Advocacy: An Analysis of Popular Code. org Videos. (2019).

[34] Ben Williamson. 2016. Political computational thinking: Policy networks, digital governance and 'learning to code'. *Critical Policy Studies* 10, 1 (2016), 39–58.

[35] Cameron Wilson. 2015. Hour of code—a record year for computer science. *ACM Inroads* 6, 1 (2015), 22–22.