

Coding to Learn and Create: New Modes of Programming for Learners Who Have Been Left Out (Work in Progress)

Colin Clark
Inclusive Design
Research Centre
OCAD University
cclark@ocadu.ca

Sepideh Shahi
Inclusive Design
Research Centre
OCAD University
sshahi@ocadu.ca

Simon Bates
Inclusive Design
Research Centre
OCAD University
sbates@ocadu.ca

Clayton Lewis
University of Colorado Boulder
clayton.lewis@colorado.edu

Abstract

While learning to code is increasingly becoming mandatory for elementary school students in many countries, learners with disabilities—especially those with complex or intersectional disabilities—are often excluded. These learners depend on assistive technologies to participate in class, communicate with family, and share with their friends. For this reason, we argue that students with disabilities can significantly benefit from the process of learning how to express themselves using computational means, and have the most at stake in becoming producers of technologies rather than simply consumers. Indeed, working with these learners raises significant questions about what coding actually entails, and the motivations and goals for learning how. The *Coding to Learn and Create* project is designing new educational coding tools that support learners with disabilities. With an emphasis on collaborative and artistic activities, we are exploring new forms of programming that support the development of life and learning skills while enabling creative expression and participation.

Introduction

Computers are, of course, everywhere around us today. Software mediates our experience with many aspects of social, school, and work life, yet its inner logics and processes can often be inscrutable to those of us on the outside. Job opportunities continue to grow for software developers and other technical workers, while the increasing role of automation in certain forms of labour, coupled with fears that so-called “artificial intelligence” may impact the role of human agency in many jobs, has invoked a feeling in many that our educational systems may well be unprepared for the future of work (Vilorio 2014, Smith & Anderson 2014). Educators and policymakers have, in response, argued for the need to teach children technical skills such as coding early in their education, as a way to prepare them for employment in an uncertain, technology-driven economy¹. In many cases, these appeals schematically link the value of “learning to code” with “STEM” (science, technology, engineering and mathematics) skills, often with a thin gloss of creativity or art applied on top. Less directly instrumental in its argument (i.e. coding == jobs), but far more wide-reaching in its belief in the importance of purely computational modes of learning is the “computational thinking” movement, which claimed that “computational thinking is a fundamental skill for everyone” and it should be as educationally essential as “reading, writing, and arithmetic... to every child’s analytical ability” (Wing 2006).

With this, perhaps, as the motivating backdrop, coding education has become an increasingly prioritized area of the educational curriculum for young children in many countries. In Canada, where several of the authors live, coding is mandatory in three provinces and has been included in the curriculum in several others. Despite this recent emphasis placed by policymakers, computer scientists, and educators on introducing students to coding at a young age, many learners are excluded from the opportunity to learn due to a lack of accessible tools and instructional methods. Students with disabilities—especially those who

¹ See, for example, the rationales of the CS4All <https://www.csforall.org/> movement in the U.S. and Canada’s CanCode program:

<https://www.canada.ca/en/innovation-science-economic-development/programs/science-technology-partnerships/cancode.html>

have cognitive, communication, or physical disabilities—are often unable to participate in today’s classroom coding activities, or are relegated to passive roles while their peers actively engage in solving problems together computationally. Yet these students, who often depend on assistive technologies to participate in class, communicate with family, and share with their friends may have the most to gain from learning how to be creators of their digital worlds, not just passive users. For example, a learner who depends on augmentative and assistive communication (AAC) software may benefit from understanding how communication boards are modelled, created, and installed, and how word prediction algorithms work, so that they can make their own custom vocabularies (e.g. to tell jokes with, or share secrets with their friends but not their parents). Or learners with mobility impairments may be able to use a coding environment connected to a robot to experiment with new forms of agency in the physical world, or learn how to give better instructions to their caretaker.

Working with students with disabilities thus raises interesting questions about what coding actually is as a practice, what it’s for, how it is manifested materially, and how it can be productively implicated with other learning and personal activities. By working with learners who are currently on the margins of computational creativity, and addressing the barriers that prevent them from engaging with computational media equally alongside their peers, new possibilities for programming languages and tools may be revealed that benefit all learners—and other programmers and users, too.

The Coding to Learn and Create Project

The Coding to Learn and Create project² (<https://codelearncreate.org>) is aiming to address the barriers to participation in coding education by students with disabilities, particularly those who are most likely to be assumed (incorrectly) to be incapable of coding, such as those with cognitive disabilities (Taylor et al. 2017). The goal of the project is to empower all learners to be creators of their digital worlds, to express themselves using code and art, and to apply these skills to other areas of learning and daily life.

While coding is too often seen instrumentally as a means to develop skills that will support future employment and career opportunities, our approach is to investigate the broader potential for coding to contribute to the development of social, daily living, and creative skills—*coding to learn* (Popat et al. 2019). The argument here is that participation in coding lessons may help support students with complex disabilities especially to develop collaborative and communication skills, strategies for problem solving, task sequencing, spatial awareness, and metacognitive skills such as those involved in giving instructions to others. Participation in coding activities also helps these learners develop a greater sense of belonging and equality with their peers in the school community. We are interested in ways that computational modes of expression can support creativity and learning, rather than simply treating the arts as a secondary concern whose role is largely to make technological concepts more appealing for kids.

The project is establishing an open, evolving repository of inclusive coding resources and activities³. These resources will be released as Open Educational Resources (OERs), and will provide educators with strategies, teaching tools, lesson plans, and techniques to help them teach more inclusively, and to adapt current coding curriculum and programming environments to better match their students’ diverse creative, social, and skill development goals. These resources will also include collaborative arts-based programming activities (such as computational drawing, music, and performances) that will provide new ways for students to learn programming while creatively engaging with their peers and the physical environment—

² The Coding to Learn and Create project is led by the Inclusive Design Research Centre at OCAD University with Bridges Canada, and is funded by a grant from Innovation, Science, and Economic Development Canada’s Accessible Technology Program.

³ An early version of this *Educator’s Toolkit* is available at <https://resources.codelearncreate.org/>

either directly or through avatars such as robots or simulations. This repository of inclusive coding OERs will be open to contribution from others, and will be designed to support reuse and adaptation.

In addition, we are co-creating, with educators, students, and their families, a new programming environment that is designed to support those who are currently unable to effectively use current coding environments. This environment will consider programming as an accessible and collaborative activity in which students may each have their own personalized interface or representation of a program, while also being able to work on shared projects and learning activities together. Additionally, teachers and students will be able to define personalized goals, rewards, and learning scaffolds that will help to support incremental skill development.

To accomplish this, the Coding to Learn and Create project is organizing a series of co-design sessions, collaborative programming workshops, and hackathons to design, test, and refine the project's learning resources and programming tools. These events act as participatory research methods that enable the evaluation, expansion and refinement of the project's deliverables in a collaborative way that is also grounded in the realities of teaching programming to very diverse students, both in the context of inclusive classrooms and congregated, disability-specialized schooling.

The following sections outline two of the key design methods we are employing throughout the project: participatory co-design with students and teachers, and continuous prototyping. We discuss ideas and strategies for how we'll engage learners in the process of creating new coding tools, describe our first prototype of an accessible turtle graphics programming environment, and summarize several areas of computation and creative expression that we will explore with our community.

Co-Designing Accessible Coding

The Motivation for Co-Design

Traditional design practices rarely engage users as participants in a substantive manner throughout the process, particularly those with extreme needs. During the initial discovery phase, researchers may engage users in activities such as interviews, focus groups, and observation sessions to better understand their needs and behaviors in a specific context. Later on in the process, they may also engage people in usability testing sessions and focus groups to get their feedback about solutions that have largely already been built. These efforts, however, do not give users the means to be directly involved in the creation and development of a solution that will inevitably impact them later.

For the Coding to Learn and Create project, we aim to take a different approach, applying a co-design process in order to more actively involve our participants as co-designers within every stage of the research, design and development process. In addition to co-design activities, we will also apply other more conventional research methods, such as surveys and interviews to further expand our reach and gather data that can complement the ideas generated throughout our co-design activities.

Planning for Co-Design

We consider that everyone is creative and capable of contributing to design and problem solving. However, when a creative action takes the form of designing, building, drawing, performing or using other creative media to express ideas, many people think (or have been told) that they do not have the proper skills, expertise, or training to contribute to such activities. This becomes even more challenging when our co-designers are from marginalized populations, such as those with different learning, cognitive and physical needs. They may feel uncomfortable sharing their ideas or be afraid of contributing thoughts that might be perceived as less good, not right, inappropriate, or irrelevant. To minimize this barrier, we will start with small-scale activities, encouraging students to participate in tractable problem-solving tasks, building up trust, and inviting them to experiment with different ways of contributing in order to find their favorite

medium. This approach helps them develop confidence and gives them—and us—ample opportunities for experimentation, determination of roles and modes of participation, and time to fine-tune the collaborative problem-solving process together.

The other aspect that may impede participation is the accessibility of the medium. Even the most confident individuals may not be able to fully participate in a co-design process if they are not provided with tools and activities that meet their needs. For example, learners with disabilities may require different ways of accessing and communicating information, such as via voice commands, gestural commands, eye gaze, communication boards, single switches or other assistive devices to be able to communicate with each other. Thus, without access to such assistive technologies—and materials that are compatible with them—they won't be able to fully participate. To this end, we are endeavouring to ensure that our tools and activities are multimodal and support a wide range of input and output methods, and are working closely with the students and their care providers to determine the appropriate tools and formats.

Co-designing with people who have cognitive, learning, or communication disabilities can be fraught with unique challenges. Depending on the context and the participants' needs, there will be situations where a student will not be able to independently participate in the co-design process, and may require assistance from or mediation by their caregivers. This may impact a student's agency and can raise questions about how truly involved they are in a co-design process. To address this issue, our team will work closely with the care providers to ensure students' insights are captured as they prefer and try to avoid overly interpreting their ideas. This requires a different approach to facilitation and engagement. For example, we have observed that inexperienced facilitators or assistants have a tendency, when asking questions of students with communication disabilities, to rush or anticipate the student's responses. Some students need more time to consider a question and respond to it, and the task of clarifying or interpreting a student's response often requires further questions and prompts. During our co-design engagements, we will provide educators and their assistants with communication strategies that give students sufficient time to think about their responses and avoid rushing them to respond or asking leading questions. This may involve tailoring the length of a session, or reducing the number of topics discussed to a more tractable scope. In addition, during the sessions themselves, we will look out for opportunities to provide students with on-the-fly adaptations that may be a better fit for their direct participation, and which will minimize the risk of misinterpreting their ideas.

At this early phase of the project, we are in the midst of reaching out to different communities and building relationships with various schools and organizations that work with students with complex learning, cognitive and physical needs. In the meantime, through collaboration with our initial group of partners and contributors, we are developing a “palette” of different co-design activities that will suit different contributors and situations, and which can be further refined as we work more closely with our co-designers and their care networks. Some of these activities are described in the next section, where we outline a process for working with students and teachers to build a more accessible and educational coding tool.

A Process for Discovering What Kids Want to Learn

To design and develop an accessible educational coding environments for kids, first we need to reassess our assumptions about coding—what it is, and how it should be taught to learners with diverse needs—and to start our work from their goals, interests, and perspectives. We will specifically recruit learners who have been marginalized or excluded from coding education due to their different needs or abilities, who will have an opportunity to share their experiences and express their visions of coding and how it could impact their lives.

Once we break out of our unquestioned assumptions about coding and consider alternative models for existing systems, we can start building new coding tools and resources. At this phase, we will work with our co-designers to design, build, and test various coding environment prototypes and tools to help actualize different models of coding that are multimodal and accessible for a wide range of needs.

Teachers and educational assistants who have relevant experience in coding will be involved throughout this process in different capacities. Since they are familiar with their students' needs and preferences, we will work with teachers directly to ensure our co-design activities and tools are accessible for all participating students. Teachers will also play an instrumental role in helping us to understand curriculum requirements, educational goals, and the practical, day-to-day challenges teaching coding to students with disabilities. They will also help bridge the communication gap and help our team to work with their students through the co-design process.

At each phase of the co-design, we are planning to use a series of individual and collaborative activities in order to engage students in a reflection and discovery process about their perception of coding, and to involve them in generating ideas that will support the design of new coding tools. These hands-on, interactive and multi-modal activities involve a mix of individual and small group participation, providing each student a chance to think on their own before collaborating with their peers. These activities build on each other to gradually introduce the students to the tools and processes used in our co-design process. The insights and ideas generated through these activities will help our team to better understand how students perceive computers and computational processes and how they prefer to learn about and engage with these processes.

The first activity in this series invites students to identify the computers they encounter in their day to day life. They are encouraged to think beyond the stereotypical concept of computers and try to identify any other forms of computation that have an impact on their lives.

A follow-up activity, inspired by Judy Robertson's work in classrooms (Robertson 2019), aims to further investigate the students' understanding of computers and how they perceive computation. Thus, they are tasked with imagining and expressing how a computer works. During this activity, students may touch upon different concepts related to coding and computational processes from their perspective.

This activity will engage students in an ideation process. Here, we encourage them to think about how they would most like to interact with those different computers they have identified in their lives. They will be provided with several challenging scenarios to help them think outside of their ordinary relationships with computers and consider some of the accessibility challenges that coders may face. For example, some of these scenarios may include giving commands to a computer that can't hear the student/a computer that can't see the student/ a computer that is out of their reach or even invisible to them, etc.

In the last activity, students are encouraged to work together to determine how they prefer to learn about computers and interacting with them. In this activity they are asked to be critical and to discuss what is not working in their current coding classes, what could be changed or improved and how they envision the future of coding education.

Once our co-designers have participated in these activities and feel more comfortable with working in a collaborative setting, we will introduce them to more advanced activities that involve interactive coding tools. At this stage, students will have a chance to:

- try out the different prototypes we are designing and developing
- sketch different user scenarios based on their personal experiences with computers to test different aspects of each prototype and identify their accessibility barriers
- share ideas about how they can change, improve, or redesign the prototypes in a way that suits them best. Students are encouraged to document their reasoning for each suggestion, and to imagine changes they would make to the prototypes.

The insights and ideas generated through these activities will help our team to better understand how students perceive computers and computational processes and how they prefer to learn about these processes. To better identify the gaps in the current computer science and coding education for kids, we are also planning to work with teachers, particularly special education educators with relevant experience.

In addition to our planned co-design engagements, we will also reach out to a broader range of teachers across Canada using a survey that has been designed and distributed to more than a hundred schools. In this survey, teachers have an opportunity to discuss challenges that they face with regard to coding education, working with students who have special needs, and opportunities they see for improving or revamping the current coding tools and curricular activities designed for their students. The survey participants can opt-in for an interview to further discuss their perspectives about a more inclusive and accessible coding education.

Continuous Prototyping as a Co-Design Method

We have started work on a prototype of a new coding environment. This prototype is not the thing that we are ultimately building, but rather, a thing to help us make the things that we will co-design with our community of students, educators, and families. Building an early prototype gives us a concrete platform on which to try out design ideas and technical approaches, and helps us when talking with others about the potential approaches and directions of the project. It serves as an invitation to participate—“this is the sort of thing that we want to build with you.” In this way, we have aimed to build something that sits somewhere between a design sketch and a product. Functional enough that we can try real interactions out (such as having a group of students program a robot to make art), but not so set that it limits our conceptions about what a coding environment could be.

Coding to Learn and Create Prototype

Commands

↑ ← → + 🗑️

Program

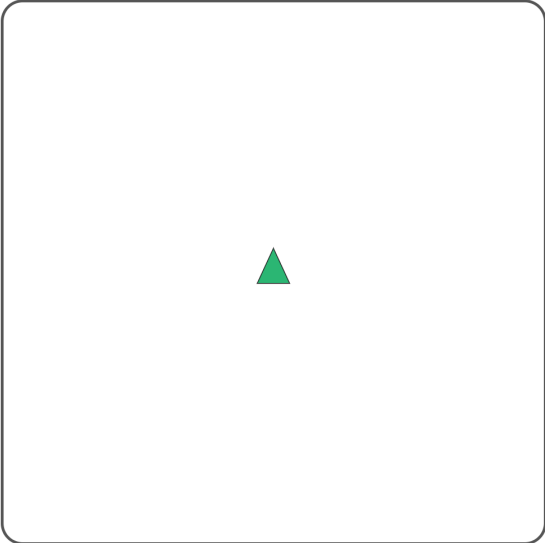
↑ ← ↑ ← ↑

← ↑ ← □ □

Program:

```
forward left forward left forward left
forward left
```

Available actions: forward, left, right.
Write your program actions in the box above and separate them by spaces. For example, to draw a square, use: [forward left forward left forward left forward left](#)



Run
Step
Restart

Connect to Dash

Connect to Sphero

Figure 1: The Coding to Learn and Create Prototype. <https://prototype.codelearncreate.org/>

The prototype is built with web technologies and the Infusion JavaScript framework (Basman, Lewis, Clark 2015). This technology platform was selected for the prototype to enable ideas to be implemented quickly and to enable us to run the prototype on the wide range of hardware devices for which web browsers are available and common in educational settings (including desktop computers, laptops, tablets, and Chromebooks). In addition to running on multiple hardware devices, web technologies have well-supported assistive technology integration and some built-in user control of web page presentation (such as zooming or setting of text size). The prototype is architected as a composition of loosely coupled parts, using Infusion’s “Inversion of Control” and “Model Relay” supports. Features of the prototype, such as editors, or robot integrations, can be added, removed, or replaced in a straightforward manner, with minimal coupling to other parts of the system. Enabling the convenient implementation of alternatives and the replacement of parts as the prototype is evolved.

As we work with educators, students, and families to co-design the new coding tools, we will assess the technology choices that we have made for the prototype and select technology platforms for the end product development that best fit the needs of our community, ensuring that what we build will work on the devices and integrate with the technologies that students are using.

Turtle Graphics and Robots

The prototype currently provides a simple turtle graphics system and a simple language consisting of three operations (currently called “actions” in the environment): move forward, turn 90 degrees to the left, and turn 90 degrees to the right. We have chosen to begin with a turtle graphics model due to its familiarity, potential for making visual art, historical usage in coding education, and compatibility with commonly available robots.

In addition to drawing a turtle on the screen, we have implemented integrations with several off-the-shelf coding robots, including the Dash robot from Wonder Workshop and the Sphero robot. Students can connect up to one of each of these robots to the prototype and when a program is run, the robot(s) will move accordingly. The robot integrations are built using the Web Bluetooth API (Web Bluetooth specification). A “Sketch Kit” add-on is available for the Dash robot that enables the attaching of a pen to Dash. With this attachment, a student can use the prototype to make drawings on paper. We are continuing to explore collaborations with robot manufacturers, with the goal of enabling our software to work with the different robots that are available for purchase and which are being used by students and in schools.

Concurrent Notations

One of the areas that we would like to explore with the project is providing multiple concurrent programming notations. Eventually, this will allow a student (or educator or family member) to select the notation, or notations, that are best for their learning needs—while still being able to collaborate with students who may prefer a different notation. In the current prototype, we offer two notations. \

The first notation we implemented is a simple text notation, where a program consists of a list of words, separated by whitespace (where each word is one of the available actions “forward”, “left”, and “right”). For example, to draw a square, the following program could be used “forward left forward left forward left forward left”. The second notation is a symbolic notation consisting of a sequence of symbol blocks, one symbol block per action. The user interface for working with the symbolic notation consists of a palette of actions (move forward, turn left, and turn right), program manipulation commands (insert space into the program, and delete a step), and a program area. Although drag and drop is typically used in block-based programming environments, this user interface pattern can be problematic for some students with physical disabilities. Rather than using drag and drop for building programs with the symbol blocks, we chose to implement an alternative two-step interface similar to the “hybrid method” described in (Milne and Ladner 2018): step 1) select the action to be performed in the palette and step 2) select the target within the program to apply the action to. We have designed the interface to have large targets (symbol program blocks and buttons) to ensure that they can be used without needing very fine motor control, or by those using assistive technologies such as eye gaze control.

Our prototype programming environment has a single underlying program model, currently consisting simply of a JavaScript array of words. Each notation is implemented as a bi-directional mapping to and from the shared program model. This enables the student to work in either notation and have the other be updated automatically. In the future, our programming environment will provide programming functionality beyond a simple sequence of actions, such as subroutines, looping, and variables. When these features are added, our programming model will of course need to become more complex to accommodate this functionality. There are many notations that we would like to try out, and the questions of a) which notations and programming models are most useful to our students and b) how to, or if we can, implement useful mappings between these notations will be topics that we will be tackling over the course of the project.

Some candidate notations for exploration include:

- An interface for controlling a robot, or turtle on-screen, that can act both as a direct ‘remote control’ and a programming environment. Where actions, such as movement, maintain their ‘liveness’ and can be used both to send commands directly to a robot and within a program.
- An interface that facilitates both direct manipulation and programming. For example, where a student could draw a shape directly, rather than with movement commands and then process their drawing programmatically, such as to repeat it with variation (for example with variation in position, rotation, colour, or line thickness). Or where programming commands are derived from the drawn shapes.
- An interface that combines programming instructions with physical control input devices. Where, for example, a parameter could be controlled by pushing or squeezing a ball.
- Dataflow programming as an alternative to imperative programming, where values within a system are related to one-another through connections and transformations.
- Notations for time-based media such as animations, music, and controlling of robots (for example a dance performance).
- Liveness within the programming environment, for example where a program is continuously running and updates to the program take effect immediately (rather than having to press a “Run” button).

Further Topics to Explore

In developing our co-design activities, there are a number of other matters we hope to explore with teachers and students, and to implement in our new educational programming environment.

Connections between coding and art and music.

One of the schools with which we are working has had great success in engaging students with physical and cognitive disabilities in art activities, some producing extraordinary results. At Beverly Public School in Toronto, artists Hien Quach and Patrick Moore developed a series of art creation processes that were personalized to the needs and abilities of their students. These processes were applied by the art class to create a series of large-scale mixed media canvases that were exhibited at the Coding to Learn and Create project launch event. These art projects suggest a potentially germane link between art and computation, where the students can learn about, perform, and create their own formalized processes and instructions for creating art in digital or physical media using code.



Figures 2 and 3: Process-based artwork created by the students at Beverly Public School.

We hope to link coding to other art activities, using the turtle graphics prototype as a starting point for some. The experience of artist Jim Johnson using a form of turtle graphics (see discussion in Repenning et al., 1998) provides inspiration. The turtle Johnson used was able to respond to lines as well as draw them, a possibility we may wish to explore.

Much has been done in producing music under program control, and we hope to explore this medium as well. Here we may be able to draw on some of our own work (`##Flocking`, `Nexus`), adapting the systems in response to the experiences of and with students in the new project.

More elements of computation.

A number of extensions to the core design of our turtle graphics prototype can be seen in prior systems, and serve to enrich the exposure to computational ideas that these systems provide. We expect to add a facility for saving and reusing sequences of commands, thus providing a simple form of subroutine. If we add the ability to give the saved sequences names we provide a form of procedural abstraction. Other extensions can provide arguments for some operations, such as a repeat action for which a number of repetitions can be specified. One can then move to support parameters for user-defined subroutines, and further to allow values calculated and stored in variables to be used as arguments. As with other aspects of the work, how much or little we explore in these directions will depend on the response of the students, both with respect to what they want to do, and with respect to what they understand and adopt.

Further abstractions.

Arguably there are other ideas about computing that are important in understanding its pervasive role in the world, but that may not play much part in practical work. One idea is that a single kind of data, numbers, or bits, can represent indefinitely many, very different kinds of things. Another is that a relatively small set of quite simple operations, on numbers or bits, that is, machine instructions, serves to emulate an enormous range of activities. Certainly the role of computers would be very different, and far more limited, if these ideas did not work out. Will our students be interested in these ideas? Can we develop the ideas in an engaging and clear way?

Back to the arts.

Moving back from computation as we know it, we think that some further computational ideas, or aspects of them, can be exercised within artistic expression. An inspiration is the loop machine, a simple device that records and plays back sounds, in such a way that a complex performance can be built up in layers, with later sounds added to ones already recorded. The loop machine demonstrates the ability to store and retrieve information, in a concrete way, while enabling new kinds of musical performance. We think it may be possible to extend this idea into the visual domain, allowing drawings to be captured, replayed, and mixed. Parallels between the two kinds of loopers may help to communicate parts of some of the ideas above, about how computers can represent and operate on different kinds of information in similar ways.

Conclusion

The Coding to Learn and Create project is designing new educational coding tools and teaching resources to support students who have been left out of learning how to code. With a particular emphasis on students with complex, intersectional disabilities such as cognitive, learning, and physical disabilities, the project

aims to reconsider the goals, motivations, and materials of how software creation is taught in light of the unique learning, creativity, and communication needs of these students. By recentring coding as a means for supporting art, life skills development, and expressive communication, the project aims to support learners with disabilities in engaging creatively and collaboratively as producers of computational media. The project's approach is rooted in community-based, open source co-design practices in which educators and students will have a significant voice in shaping the direction and outcomes that we create. We welcome participants from a broad range of backgrounds, including computer scientists, artists, educators, and researchers.

References

Basman, Antranig, Colin Clark, and Clayton Lewis. "Harmonious Authorship from Different Representations (Work in Progress)." In *Proc. PPIG 2015 Psychology of Programming Annual Conference. Bournemouth, England, 15th-17th July*. 2015. [PDF]

Milne, Lauren R., and Richard E. Ladner. 2018. *Blocks4All: Overcoming Accessibility Barriers to Blocks Programming for Children with Visual Impairments*. In Proceedings of the ACM Conference on human factors in computing systems (CHI '18). ACM, New York, NY, USA. [PDF]

Popat, Shahira, and Louise Starkey. "Learning to code or coding to learn? A systematic review." *Computers & Education* 128 (2019): 365-376.

Repenning, A., Ioannidou, A., & Ambach, J. (1998). Learn to communicate and communicate to learn. *Journal of Interactive Media in Education*, 1998(2). [PDF]

Robertson, Judy. "Answering Children's Questions About Computers." *Communications of the ACM*, January 2019, Vol. 62 No. 1, Pages 8-9. <https://cacm.acm.org/magazines/2019/1/233512-answering-childrens-questions-about-computers/fulltext>

Smith, Aaron, and Janna Anderson. "AI, Robotics, and the Future of Jobs." *Pew Research Center* 6 (2014).

Taylor, Matthew S., Eleazar Vasquez, and Claire Donehower. "Computer programming with early elementary students with Down syndrome." *Journal of Special Education Technology* 32, no. 3 (2017): 149-159.

Vilorio, Dennis. "STEM 101: Intro to tomorrow's jobs." *Occupational Outlook Quarterly* 58, no. 1 (2014): 2-12.

Web Bluetooth specification, Web Bluetooth Community Group <https://webbluetoothcg.github.io/web-bluetooth/>

Wing, Jeannette M. "Computational thinking." *Communications of the ACM* 49, no. 3 (2006): 33-35.