# Project Bloks:

# designing a development platform for tangible programming for children

**Authors**

Paulo Blikstein (Stanford University, USA), Arnan Sipitakiat (Chiang Mai University, Thailand), Jayme Goldstein (Google), João Wilbert (Google), Maggie Johnson (Google), Steve Vranakis (Google), Zebedee Pedersen (Google), Will Carey (IDEO).

## Abstract

In the 1960s, Seymour Papert advocated that all children should learn computer programming. Today, finally, millions of students are learning it, but there is a lot of work to be done to truly democratize this new literacy. Project Bloks is a research project with the goal of creating a platform to enable designers to create new and creative tangible coding languages and kits for children. Ultimately we want to offer a powerful and open hardware and software platform that makes it easy to invent novel ways for children to learn how to code, using new form factors, metaphors, and knowledge domains.


## Coding: more than a job skill, it is a new literacy for the 21st century

In 1972, two researchers working at MIT, Seymour Papert and Cynthia Solomon, published "21 Things to Do with a Computer"—a visionary paper that foresaw the coding-for-children movement, the maker movement, and most of the creative uses of educational technologies that we see today. What enabled them to see so far into the future? They assumed that the most interesting use of technology in education would not be electronic teachers, or the machine teaching the child. Papert and Solomon envisioned the child as teacher of the machine. In other words, they tried to imagine what would happen if children could use computers to express ideas, construct things, and create inventions, as opposed to simply using technology to receive pre-packaged information. More recently, Papert's collaborators, such as Andy diSessa, contradicted another well-established conception, stating that computer programming is not simply a job skill, but a foundational literacy for everyone to learn. In the same way that we don't teach music in schools for students to become professional violinists, or English for students to get jobs in journalism, we should not teach programming for children to get programming jobs: we should do so for them to acquire a new way of thinking and seeing the world. It should be done because we want them to be producers of

technology and not mere consumers. Coding is the kind of skill that you cannot "unlearn"—once you discover it, the way you see the world will never be the same. And that is why we should care so much about teaching programming and computational literacy to all: *not as a job skill, but as a thinking skill.*

## How are we bringing computational thinking to children?

One of the most popular ways to bring coding to children has been physical computing and robotics, since many children enjoy partaking in these kinds of activities already. Researchers have been creating these kinds of platforms for decades with great success, enabling students to build inventions, robots, and devices, and then program them. One of the first were the Programmable Bricks, created in the nineties at MIT, which inspired the popular LEGO® Mindstorms robotics kit. This work also inspired a series of other platforms over the years, such as the Cricket, Pico Cricket, GoGo Board, Wiring, Arduino, and Phidgets. More recently, new form factors, modes of interaction, and materials have been proposed: the Lilypad kit enables users to create and program e-textiles, Topobo allows students to program by example instead of typing code, and platforms such as the MIT Tangible Programming Bricks, Cubelets, MOSS, and LittleBits enable programming by simply assembling physical blocks together, without a computer. A variation of this idea was tried at Tufts University with Tern, a system in which each physical wooden block corresponded to a programming command. The idea behind Tern was not to engage children in physically building a robot or device, but to engage them in writing a program by manipulating tangible blocks. The individual blocks had optical marks that were read by an overhead camera, which were compiled by a computer and sent to a physical robot nearby. These types of systems are typically used with children as young as five years old, and have many descendants, such as Kibo and more recently, Cubetto.

## A design space in intense transformation, and many opportunities for new ideas

In the last five years, this design space has been in intense transformation because of three factors. First, coding went mainstream, due to high-profile private and public initiatives and campaigns to popularize it. This popularity made research labs, design firms, creative educators and engineers turn their minds to the creation of new platforms for coding. Second, the popularization of crowdsourcing platforms gave those groups a viable way to fund and commercialize their ideas. Finally, new technologies such as low-power wireless communication, low-cost rapid prototyping, and new types of microcontrollers and microprocessors expanded the design space in unprecedented ways. It became possible to offer higher levels of abstraction using more sophisticated hardware and software and a more creative mix of *on and off screen* interactions. Those technologies also made it feasible to "talk" to all kinds of everyday devices using *Internet of Things (IoT)* technologies, and new form factors and types of design emerged, making programming accessible for many new audiences, younger children in particular.

However, this design explosion had some shortcomings too, such as the emergence of many single purpose, proprietary and often expensive designs that could not communicate with one another. Different groups were working on their own designs and rarely shared the same platform, design principles, or technologies. Many of these products were meant to be commercialized for individuals, and because they were not primarily designed for schools and formal education, using them in classrooms diminished the focus on collaboration, classroom dynamics, and generating low-cost designs. Therefore, despite the explosion of creativity and new products, there are wide open design opportunities and technological needs within the tangible programming space.

One urgent need is easily available—open source and extensible platforms, instead of more one-off products, which would provide the research and design communities with a common "language" for development. The

market is saturated with too many incompatible designs. A *platform* would also address the problem of increasingly complex hardware and software, which is a barrier for new designers, especially if they want to focus on educational design, rather than the technical aspects. Google's Blockly—a platform for the creation of on-screen, block-based languages—is a great example of how such platforms can empower the design community. There are many opportunities for new conceptual designs as well. With better platforms, the design community could find even more powerful uses of tangibility and physicality, so that physical programming languages are not just transpositions of on-screen languages into the physical world. There is also work to be done on making tangible programming tools compatible with the dynamics and social protocols of schools. For example, offering good supports for collaborative teamwork, and making it easier for teachers to orchestrate sessions in their classrooms.

## Project Bloks

Our contribution to this design space was born from our insights into the value of a platform-based approach, demonstrated by the success story of Blockly. The goal is to follow this model, helping the community build their own languages and products. Project Bloks aims to be to tangible programming what Blockly is to on-screen block programming.

The Project Bloks system enables educators and makers to create new tangible programming languages without having to deal with low-level technical details. We want to create a platform that will allow the emergence of lots of new designs based on Papert's and Resnick's idea of low-thresholds, high ceilings, and wide walls. We should make it easy for novices to get started, but possible to build complex and diverse creations once children's confidence develops—the tool grows with the child as opposed to being a disposable one-off design. That's how Project Bloks was born—designed in partnership between Google Research, Google Creative Lab, IDEO, Paulo Blikstein (Stanford), and with contributions from Arnan Sipitakiat (Chiang Mai University).

We want to allow designers, educators, and makers to spend more time experimenting with form factors, materials, and feedback channels (haptic, visual, and auditory) rather than getting stuck in the technology. In other words, we want to enable them to be more creative with language paradigms and abstractions rather than having to spend most of their time thinking about the technical side of things. The goal of Project Bloks is to offer a powerful, open hardware and software platform that makes it easy to implement sophisticated programming ideas such as functions, recursion, and complex control structures. It talks to the child's world by allowing seamless communication with external devices and sensors, making use of today's standard protocols and providing modules that are based on inexpensive, extensible, and customizable components. It uses open frame electronics, which allows the creation of different shells and packaging.

One of the key innovations of Project Bloks is that its architecture uses three types of circuit board module: a Brain Board, Base Boards, and pucks. The Base Boards contain all the electronics and have connectors on all sides, so they can be put together in a number of configurations, allowing for different programming flows (vertical, horizontal, or even a grid in two dimensions). Each is fitted with a haptic motor and LEDs that can be used to give users real-time feedback. Pucks go on top of the Base Boards, and are incredibly cheap and easy to make. They use an inexpensive, capacitive ID system which does not require active electronic components. Pucks can be made with any material (children can make their own too!) and they can contain different mechanical triggers ("physical" faders, buttons, sliders, dials), enabling different user experiences and interfaces to be created. The Brain Board is the processing unit of the system. It receives the commands from the Base Boards, interprets those commands, and sends them via WiFi or Bluetooth to a connected device. This architecture enables designers to focus their energies on making the most of tangibility and physicality.

## Trying the Project Bloks system with children

To better understand how the features of the system might enable new ways of interacting with tangibles and coding, we designed a "Coding Kit". This kit was our first attempt to create a finished set of blocks for teaching programming. Over several months, we worked with more than 150 children to understand, from the ground up, how they played with physical code.

We have run several studies with more planned over the next few months. Our initial qualitative user studies showed how each of the design features performed in the real world when our Coding Kit was used by pairs of children aged 5–8 years old. In general, we were positively surprised at how quickly children understood the rules of engagement. First, we noticed that the physical cues (e.g. magnetic snap fits, matching directional connectors, mapping of form and function of blocks) enabled children to get comfortable with the tools without asking for too much help. Here we took a page from software such as Scratch, which allows children to get started with almost no instruction. Tools that require too much initial onboarding often have the unintended consequence of disengaging students if they fail to get the simplest operations done. With this Coding Kit, within the first minute, children could figure out by themselves how to connect the blocks and how to orient them. Within the first five minutes, they also understood the order of execution (left to right), and most of them got the idea of interchangeable pucks, removing and replacing them easily on the Base Boards. The magnetic, directional snap-fits successfully guided users to connect particular blocks to each other, in the right way.

Even with all those successful results, we noticed opportunities for improvement. For example, some children had trouble understanding how to use the repeat blocks, using them in the wrong order or placing repeat commands after, rather than inside the repeat blocks. Since the idea of Project Bloks is to be a platform and not the perfect programming language, it immediately got us thinking about how the community might help us generate a better design.

Debugging is a critically important (and sometimes frustrating) aspect of coding. Helping kids find errors in their programs is one of the hardest problems interaction designers struggle with. One of the major learnings from our studies is that more types of immediate feedback can be a key element in creating a positive experience for children. Our system includes a number of debugging tools that help children see problems with their code before they execute it. For example, blocks blink red when there is an error that will prevent compiling and green when they are being executed. We observed that the children found the problematic block within seconds, and also understood, conceptually, the difference between a compilation and a runtime error. They also immediately developed a productive routine of observing the code and the robot with "debugging eyes".

One important question for us was how children would use functions. It is difficult to implement functions with tangibles, and even harder to make them easy to use. In our studies, the children demonstrated impressive levels of understanding concerning what functions can do and how to use them. With our Coding Kit, children can program a blank puck to receive a "copy" of a set of blocks, and then draw on top of it, making it not only mnemonic but also very personal. Children were quick to understand that their new block "has the same thing from there [their old program]," and could be reused. For example, when we asked them to create a flower with multiple petals, most of the children had no difficulty understanding, with very little prompting, that they did not have enough blocks to build the flower without encapsulating parts of the code in the blank puck. With this understanding, they were able to immediately begin manipulating the new function puck in the same way they did other, pre-built pucks.

## Conclusion

These results and observations, albeit from pilot studies, give us some assurance that the technical affordances of the platform will allow the community to generate engaging products that will get more kids falling in love with coding and exploring deep ideas in computational

thinking. Some of the technical and design advances of Project Bloks will hopefully make it easier to introduce children to sophisticated concepts in programming within a variety of domains and contexts, such as music, arts, science, and other elements of everyday life.

We hope that this is just the beginning for Project Bloks. In the future, for example, we could freely provide all the hardware specifications online, complete with design files so all parts can be made or 3D printed anywhere. We also want to extend the types of activities that are possible with the platform, adding pre-installed integrations to toys, existing physical computing platforms, IoT devices, connected home appliances, and other devices and services that operate via the web. We also envision allowing custom pucks to be created/exchanged between people, using a customized, easy-to-use puck creator environment with which users could create their own pucks or extend existing kits. Ultimately this could jumpstart 'Built with Bloks' or 'Taught with Bloks' communities, showcasing experiments from makers, teachers, and enthusiasts.

Research and design for children is our passion. Designing the Project Bloks system was, above all, an exercise to demonstrate how much children can accomplish with the right tools, how much they can learn when they are not told what to do, and how much reward exploration can bring them.

The vision of Seymour Papert 50 years ago was a powerful one: children will program the computer. It won't be the other way around.

## References and inspiration

The platforms, researchers, and designers that inspired our work are here: http://projectbloks.withgoogle.com/research/#algoblocks