Contents lists available at ScienceDirect

International Journal of Child-Computer Interaction

journal homepage: www.elsevier.com/locate/ijcci

Research paper

ScratchJr design in practice: Low floor, high ceiling

Jessica C. Blake-West [*], Marina U. Bers

Lynch School of Education and Human Development, Boston College, Chestnut Hill, MA, USA

A R T I C L E   I N F O

Article history:
Received 21 March 2023
Received in revised form 7 June 2023
Accepted 19 June 2023
Available online 24 June 2023

Keywords:
ScratchJr
Computer science education
Programming
Technologically rich learning environment
Early childhood

A B S T R A C T

The demand for developmentally appropriate tools and learning environments for early childhood computer science education is greater than ever. One of the most widely used introductory coding environments designed for children ages 5–7 is ScratchJr (Bers & Resnick, 2015; Flannery et al., 2013), which has over 40 million users worldwide. ScratchJr was designed as a "low floor, high ceiling" learning environment — meaning that it is accessible to novice users while also allowing more experienced users to grow their knowledge. In this paper, we evaluate how the "low floor, high ceiling" design of ScratchJr is received in practice through looking at the programming performance of users at different ages, timepoints, and experience levels. We find that the youngest, and most novice users were able to engage with the app to some extent with no instruction, but engagement was optimized with curricular support. Additionally, we found that the oldest, and most experienced users still had room for growth and discovery in the app. We conclude that the design decisions in ScratchJr such as iconography, block variation, and open-endedness of the environment creates a welcoming and engaging experience for a wide range of users, both children and adults.

1. Introduction

Over the past two decades, the development of technology and its integration into everyday life has dramatically changed how citizens engage with society. Not only are computer science skills a growing need in the workforce (Code.org, 2018; Fayer et al., 2017), but technological skills have become deeply ingrained in people's ability to communicate, learn, and express themselves. Exposure to new technologies begins early in a child's life. A national survey found that over 85% of parents with children under the age of 6 reported that their child had some degree of technology exposure (Erikson Institute, 2016).

As the prevalence of technology in young children's lives continues to grow, the demand for developmentally appropriate technological learning environments is greater than ever. Traditionally, computer science education happens in high school and college, by which time there are achievement gaps in STEM fields, which tend to be split along gender, race, and socioeconomic lines (Betancur et al., 2018; National Center for Science Engineering Statistics, 2023). In the past decade, however, there has been a push to introduce computer science to earlier ages and make programming accessible to novices of any age with environments such as ScratchJr (Bers & Resnick, 2015; Portelance et al., 2015), Code.org, Beebot robot (www.terrapinlogo.com), KIBO robotics (Bers et al., 2014; www.kinderlabrobotics.com), etc. Bringing computer science education to early childhood can mitigate achievement gaps (Sullivan & Bers, 2016) and help cultivate a positive relationship with technology (Bers et al., 2012).

One of the most widely used introductory coding environments explicitly designed for children ages 5–7 is ScratchJr (Bers & Resnick, 2015; Flannery et al., 2013) which is available on iPads and Android devices, as well as Chromebooks (as of 2021) and iPhones (as of 2023). The app was created through a National Science Foundation grant that funded a partnership between the DevTech Research Group directed by Marina Bers, and the LifeLong Kindergarten group directed by Mitchel Resnick. The two groups together, with Paula Bonta from the Playful Invention Company, began the work of designing a coding environment that is usable for early childhood education: ScratchJr.

ScratchJr is inspired and adapted from Scratch, created by the LifeLong Kindergarten group at MIT Media Lab (Maloney et al., 2010). Scratch is designed to make programming playful and creative so programmers at all levels can orient themselves (Myers & Stylos, 2016). In contrast to other languages, which are syntactically difficult, Scratch offers visual programming blocks, the ability to create interactive art and stories, and the output of animations (Resnick et al., 2009).

Despite the inviting nature of Scratch, it has a large instruction set and relies on complex concepts and text labels to communicate ideas — all of which are barriers for most children between the ages of 5–7. Despite the fact that research has shown that

[*] Correspondence to: Carney Hall at Boston College Chestnut Hill Campus, Attn: Room 306, 281 Beacon St, Chestnut Hill, MA 02467, USA.
E-mail addresses: jessica.blake-west@bc.edu (J.C. Blake-West), marina.bers@bc.edu (M.U. Bers).

https://doi.org/10.1016/j.ijcci.2023.100601
2212-8689/© 2023 Elsevier B.V. All rights reserved.

children as young as 5 years old can master fundamental computer science concepts such as sequencing, logic, and cause and effect when working with appropriate tools, there is a need to make developmentally appropriate programming environments (Bers & Sullivan, 2019; Strawhacker et al., 2015). Thus arose the motivation for creating ScratchJr: an environment designed specifically to make programming accessible to young children.

As of February 2023, ScratchJr has been used by over 40 million people, in 192 countries. 167 million projects have been created, 275 million projects have been edited, and 7 million projects have been shared. The DevTech Research Group has developed over 60 hours of ScratchJr curricula and hundreds of other educators have incorporated ScratchJr into the classroom and posted their projects and activities on the web through ScratchJr Connect (Blake-West & Bers, 2023). ScratchJr is maintained by the DevTech Research Group, now at Boston College, and the Scratch Foundation which generously provides funding so ScratchJr can remain a free programming environment for all children across the world.

The design of ScratchJr was researched, tested, and iterated upon in order to make it suitable for novice programmers (Ben-Ari, 1998; Kelleher & Pausch, 2003; McKeithen et al., 1981; Norman & Draper, 1986) and developmentally appropriate for its intended user age group of 5–7 (Rader et al., 1997). Four themes were identified early on by the design team (Flannery et al., 2013) regarding the design of the interface and the user experience:

> "1) Low Floor and (Appropriately) High Ceiling: Make it easy to get started with ScratchJr programming. Provide room to grow with concepts varying in complexity, but keep the tool manageable for the range of users.
>
> 2) Wide Walls: Allow many pathways and styles of exploration, creation, [expression] and learning.
>
> 3) Tinkerability: Make it easy to incrementally build up creations and knowledge by experimenting with new ideas and features.
>
> 4) Conviviality: Make the interface feel friendly, joyful, inviting, and playful, with a positive spirit of exploration and learning"

Each of these design themes informed different features and affordances of the coding environment, however little work has been done to date to investigate the effectiveness of these design decisions since ScratchJr's development back in 2014. Now, after over ten years and 40 million users, we have the opportunity to draw on ScratchJr user data to investigate whether the design themes identified have been successfully implemented to create a developmentally appropriate coding environment.

Although user data can be explored in different ways, for the scope of this paper, we will examine the "low floor, high ceiling" nature of ScratchJr through looking at programming performance of users at different ages, timepoints, and experience levels. This approach will allow us to investigate whether ScratchJr has an appropriately low barrier for entry for the age group of 5–7, and whether it affords growth for more experienced users.

Previous work has explored the flexible nature of other programming environments, specifically those with a low barrier for entry. Environments such as Logo (Papert, 1980), Boxer (Di Sessa, 1985), Squeak (Ingalls et al., 1997), NetLogo (Wilensky, 1999), and Scratch (Resnick et al., 2009) have been found to be easily accessible to novices, while also engaging users in complex ways including the use of loops, conditionals, and synchronization (Berland et al., 2013; Maloney et al., 2008). These environments provide users with a simple syntax (Guzdial, 2004) and immediate and visible feedback (Maloney et al., 2010) to support users in the acquisition of design thinking and problem-solving skills necessary in complex programming tasks (Soloway, 1986), without being hindered by syntax errors (Maloney et al., 2008). Although these environments were designed for children, the target age range is for those who are already literate, which is around age eight and older. This study is the first to evaluate whether the ScratchJr programming environment is flexible enough to meaningfully engage a wide range of users of varying skill levels, ages, developmental needs and academic stages.

We examined the experiences of students from kindergarten to second grade as well as their teachers using the ScratchJr app. Adults are included in this investigation because, when considering the user experience of an early childhood learning environment, children are not the only users. It can be assumed that an adult, such as a teacher or parent, will be engaging with the app along with the child. The goal behind this investigation is not to show that ScratchJr has *no* floor and *no* ceiling for engagement, but rather to investigate where those boundaries to engagement fall along age, experience and skill levels — and whether they are appropriate for the target user group(s) of ScratchJr.

To investigate whether ScratchJr has a "low floor" entry, we asked novice users to navigate the ScratchJr interface as we recorded their answers as successful or unsuccessful. We then analyzed their success rates to determine whether there was a floor effect in the data set, which would suggest that ScratchJr does have some entry level barriers. We hypothesized that there would not be a floor effect for the children in this study, as they fall within the expected age range of ScratchJr: 5–7 years old, thus showing that ScratchJr is readily accessible to children as young as 5 years old. Similarly, to investigate whether ScratchJr has a "high ceiling", we asked the same age groups to engage with increasingly complex programming activities, after they participated in a curricular intervention. We then examined whether those success rates had a ceiling effect which would suggest that ScratchJr only affords for limited growth and users may outgrow the tool after experience and/or instruction. We hypothesized that children K-2 would not have a ceiling effect, suggesting that ScratchJr is a flexible educational environment that caters to a wide range of abilities and learning paces. For adults, however, we hypothesized that there would likely be a ceiling effect, for a variety of reasons including reading abilities, learning pace, and ability to grasp abstract concepts — suggesting that ScratchJr's ceiling is appropriately high for young children, but not limitless.

2. Background

2.1. Low floor high ceiling in technologically rich learning environments

The concept of "low floor, high ceiling (LFHC)" in the context of learning technologies was first introduced by Seymour Papert in the 1970s within the framework of Constructionism. Constructionism is a learning theory which advocates for student-led exploration and project based learning (Harel & Papert, 1991; Papert, 1980), as well as flexible learning environments which can accommodate this type of learning through the use of microworlds (Bers, 2020). Papert coined the term LFHC when describing his design principle for the programming language Logo — the first programming language to be accessible to young children while also being engaging for adults (Papert, 1980). A "low floor, high ceiling (LFHC)" environment means that the environment is accessible to a wide range of skill levels. At the core of Papert's work was enforcing the idea that "everyone can get started, and everyone can get stuck", regardless of age and ability (NRICH Team, 2019). This concept affords people to be working

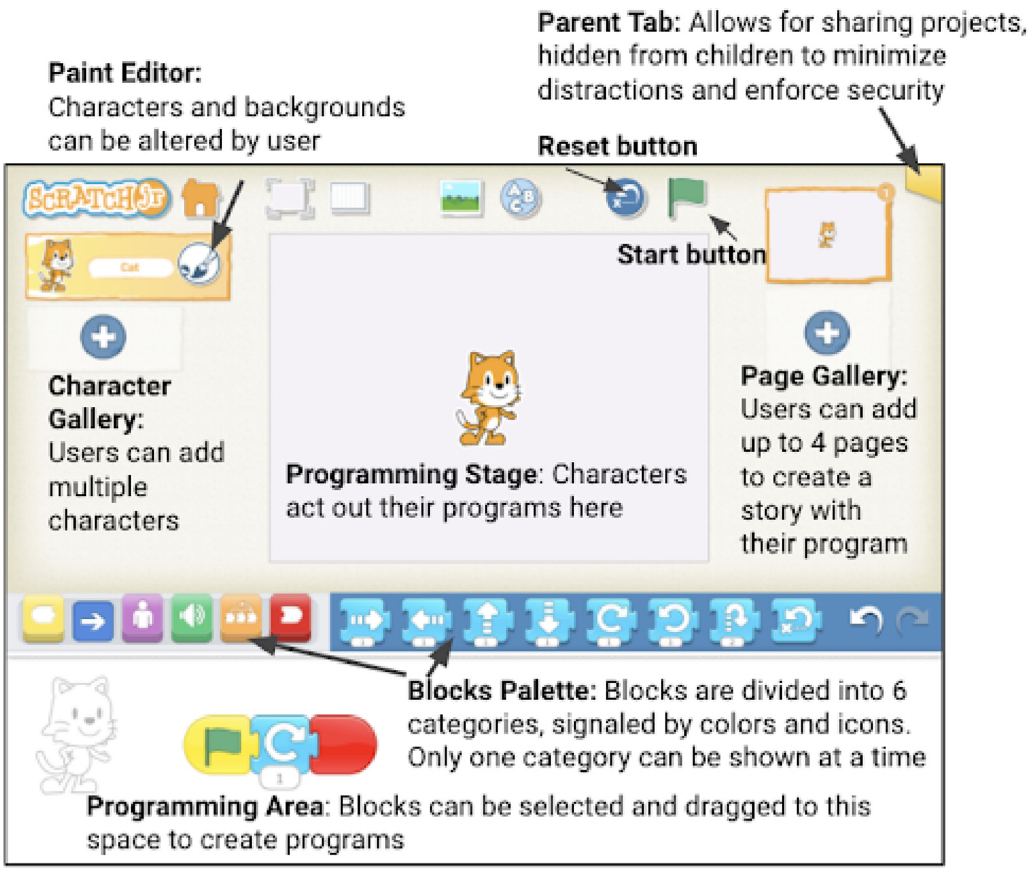

Paint Editor:
Characters and backgrounds can be altered by user
Parent Tab: Allows for sharing projects, hidden from children to minimize distractions and enforce security
Reset button
Start button
Character Gallery:
Users can add multiple characters
Programming Stage: Characters act out their programs here
Page Gallery:
Users can add up to 4 pages to create a story with their program
Blocks Palette: Blocks are divided into 6 categories, signaled by colors and icons. Only one category can be shown at a time
Programming Area: Blocks can be selected and dragged to this space to create programs

Fig. 1. ScratchJr interface.
Note. This figure showcases key elements of the interface, most notably: the programming stage, where characters act out programs, the blocks palette: where blocks are divided into six categories, with only one category being displayed at a time, and the programming area: where blocks are dragged from the blocks palette and connected to create programs.

at their zone of proximal development (Vygotsky, 1978) because they are given the space to grow when supported and scaffolded. Through this lens, we can understand the LFHC design principle as a pathway to universal design and inclusive education. As stated by the NRICH team at University of Cambridge, "[LFHC] tasks allow learners to demonstrate what they can do, rather than what they cannot... When the ceiling is raised it can be surprising what heights learners can achieve" (NRICH Team, 2019).

Similarly, other work (Robins et al., 2003) shows that novice and expert programmers interact with a programming environment and programming tasks very differently and thus an environment must offer entry points and expansions that cater to the needs of all types of interaction. Shertz and Weiser (1981) found that for novice programmers, problem solving relies on more superficial cues of the environment or syntax, whereas expert programmers have automated their recognition of syntax and take on more abstract, multi-step processes (Wiedenbeck, 1985). ScratchJr's open ended design makes both types of interaction possible.

2.2. *ScratchJr as a low floor high ceiling learning environment*

2.2.1. *ScratchJr interface and experience*

The ScratchJr interface was designed using the LFHC design principles. Fig. 1 highlights the key features of the app's project editor interface. Most importantly, Fig. 1 explains (1) the programming area, where blocks can be dragged and snapped together to create programs, (2) the blocks menu and palette, where the blocks can be selected from, and (3) the programming stage, where the animated outputs of the programs occur.

2.2.2. *Low floor in ScratchJr*

ScratchJr is designed to be intuitive enough so that children as young as 5 years old can get started without any formal instruction. In ScratchJr, a novice user interacting near the "floor" of the coding environment may only be able to enter the environment (creating a ScratchJr project to enter the editor) and understand how to navigate some aspects of the interface, but not actually get as far as making a program. For example, they can select blocks from the block menu, drag a block to the programming area, add and/or customize a character or background, or recognize the symbols shown on each programming block. It is not assumed that all these actions will come easily upon first exposure — another design principle of ScratchJr is the ability to explore and tinker. However, there is a certain level of engagement needed for further exploration, specifically opening the project editor, accessing a customizable feature and interpreting and interacting with programming blocks.

Given that ScratchJr is a programming environment, the emphasis on the programming blocks is conveyed through a centrally located, brightly colored blocks palette composed of icon-based programming blocks. The largest barrier to young children on Scratch was the platform's reliance on words. Thus a main design choice for ScratchJr was eliminating the reliance on text and creating simple commands with universal symbols representing its functions (Strawhacker et al., 2015), which has been found to improve interface usability for low-literacy users (Medhi et al., 2011). There are other ways to meaningfully engage with the educational tool in addition to programming, including customization and art features. This is particularly important for creating a lower threshold for entry.

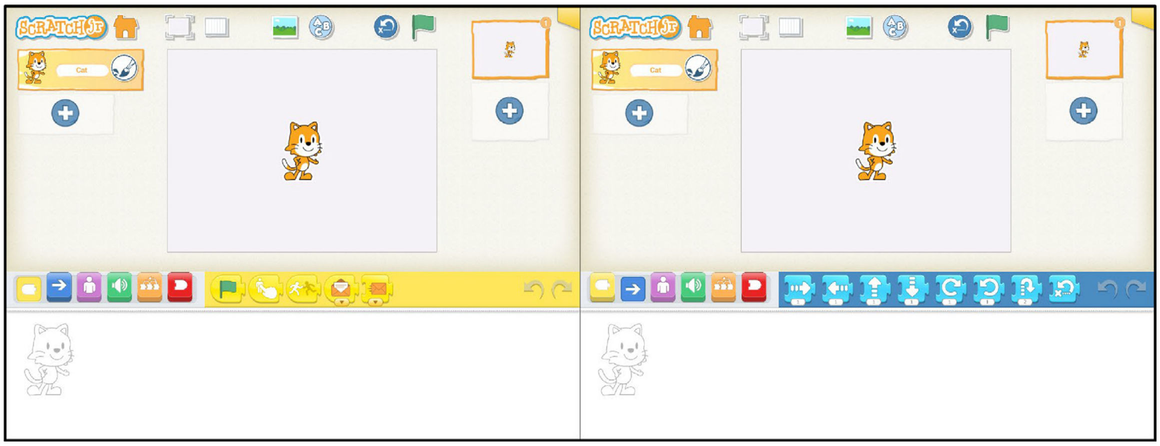


Fig. 2. Block palettes on ScratchJr.
Note. This figure showcases different block categories. On the left, the yellow Start blocks are selected, and on the right the blue Motion blocks are selected.

2.2.3. *High ceiling in ScratchJr*

ScratchJr is an open-ended coding "playground" (Bers, 2020), in which users can build on their projects with different block types, combinations and extensions, and encounter opportunities for high-level thinking and problem solving, characteristic of more experienced programmers (Wiedenbeck, 1985). The more complex blocks include the orange Control blocks — which do not have an immediate action when used in isolation, but rather affect (or "control") other blocks when used in combination. These blocks include the Repeat Loop, the Set Speed blocks, and the Wait block, all of which offer users an opportunity to engage with more abstract programming concepts. Additionally, the different types of trigger blocks begin to introduce users to varying levels of complex conditionals and events. Finally, ScratchJr allows for program combinations. Rather than limiting characters to a certain number or length of programs, users are able to create programs that run in parallel, trigger one another, or stop one another. By offering a wide range of blocks, and the ability to run multiple programs at a time, ScratchJr offers an almost infinite set of possible block combinations and outputs, giving more experienced users opportunities to push themselves.

ScratchJr affords for a high ceiling through the open-ended experience of the app and the expansiveness of block complexity rather than the interface design. However, with open-endedness comes the risk that some more users may need further support and guidance than is provided (Clarke-Midura et al., 2019). To address this, ScratchJr has interface design choices which aim to guide the user experience, such as streamlining the presentation of the blocks. By focusing the users attention on a select portion of commands, ScratchJr's design retains an *appropriately* high ceiling without overwhelming beginners. The blocks palette displays up to eight blocks at a time, specific to a certain category of functionality (Start/Trigger, Motion, Looks, Sound, Control, and End) (see Fig. 2) to help mitigate the more complex functions from being confusing or intimidating to more novice users.

3. Study description

In this paper, we investigate the user experiences of both novice and expert users to understand if the "low floor, high ceiling" design held up in practice. In order to investigate the LFHC hypothesis, we utilized the Coding Stages Assessment (CSA), a validated, age appropriate, task based ScratchJr assessment designed to measure users' abilities to engage with coding on ScratchJr (de Ruiter & Bers, 2021), and used in previous research studies (Bers et al., 2023). The CSA is both authentic and adaptive in order to cater to limited knowledge transfer skills and attention spans, respectively. The CSA exposes users to all different possible interactions with ScratchJr, for example: opening the editor, interacting or interpreting blocks, and navigating the interface for customization options. In addition, it invites users to complete highly complex and multi-step programming processes, probing the level of complexity to which the users are able to engage.

CSA is composed of five stages, each composed of six ScratchJr tasks, developed upon the coding stages theoretical framework (Bers, 2019). For example, the Emergent stage tasks begin with basic interface interactions (Table 1) and New Knowledge stage tasks progress to complex, multi-step program tasks (Table 2). Each of the indicators listed in Tables 1 and 2 were identified (Bers, 2019) and validated (de Ruiter & Bers, 2021) for children ages 4–8 interacting with ScratchJr. This paper explores the baseline CSA scores of children and teachers to understand if the ScratchJr coding environment affords low levels of coding knowledge, and the scores after intervention to understand if it is possible to reach a ceiling effect in the types of programs that can be created.

In this paper we are operating under the assumption that the floor of the CSA performance is indicative of the floor of ScratchJr interactions and thus will inform us whether the environment itself has a low threshold for entry. To investigate the ceiling of ScratchJr, we are operating under the assumption that the ceiling of the CSA scores is likely lower than the actual ceiling of the possibilities that ScratchJr offers, since there are many possible combinations of interactions, not all of which can be measured within the 30 tasks presented in the assessment.

3.1. *Identifying floor and ceiling effects*

To identify floor and ceiling effects, we examined the proportion of users who scored the lowest possible score(s) as well as the proportion of users who scored the highest possible score(s). To determine if proportions are indicative of a true floor or ceiling effect, and thus suggesting a barrier for entry or a lack of flexibility for growth, we researched prior work done to identify floor and ceiling effects in testing (McHorney & Tarlov, 1995) and found that setting a threshold of 15% for the lowest and highest possible scores in an assessment can be assumed to indicate a floor and ceiling effect, respectively (McHorney & Tarlov, 1995). To justify this threshold, specifically threshold for entry, we conducted a qualitative investigation on a small proportion of the assessments which were video recorded, which allowed us to have a more in-depth understanding of what a score of

Table 1
Emergent tasks: Indicators of entry level skills necessary for engaging with the app.

Tasks	What aspect of interacting with ScratchJr is being tested?
1.1: Please open the ScratchJr app and start a new project.	• Basic knowledge of apps • Navigating home screen interface
1.2: Can you make a program with these three blocks?	• Navigating interface — specifically selecting from the block menu and dragging blocks into the programming area • Shapes of the blocks and order matters
1.3: Now play the program. What is Cat doing and why?	• Understanding their role as programmer on the app • Navigating interface to play program
1.4: Look at this program. Which block is making Cat get bigger?	• Iconography on blocks and understanding representation
1.5: Can you add a friend for Cat? It can be any character.	• Navigating interface - how to add multiple characters
1.6: Can you add a background for Cat? It can be any background.	• Navigating interface - how to customize the background

Table 2
New knowledge tasks: Indicators of the upper most boundary of complexity which ScratchJr Affords.

Tasks	What aspect of interacting with ScratchJr is being tested?
4.1: Watch this video. Can you program Cat to move forever to the side and at the same time, jump when you tap on it?	• Are they able to parallel program (run multiple programs simultaneously)
4.2: Watch this video. Can you program Cat to move like this? (move in a smooth diagonal)	• Are they able to parallel program (run multiple programs simultaneously) with the understanding that motions can be combined when running simultaneously
4.3: How many times will Cat make a pop noise?	• Use the repeat loop to create nested loop programs
4.4: Using Message blocks, can you program Cat to jump, then dog to jump, then rabbit?	• Fully utilize the broadcasting function (use the multiple color messages intentionally and accurately)
4.5: Here are the programs for Cat doing a cartwheel. Can you program Dog to bump into Cat and then cat will do the cartwheel?	• Combining multiple advanced functions on the app to create a complex program. In this case, Start on Bump trigger block (concept of conditionals) and parallel programming (run multiple programs simultaneously)
4.6: Can you program Dog to send a message to Cat to go to the next page?	• Combining multiple advanced functions on the app to create a complex program. In this case, broadcasting and multi-page projects.

0 or 1 looks like. We found that a typical "floor" level user usually is not familiar with opening apps generally, and thus needs assistance getting started (opening the app) but then is able to recognize the meanings of the blocks, even if they are not able to drag them to the correct area. In considering what floor-level engagement looks like, we determined that this level of performance accurately portrays the lowest possible amount of engagement with the app that can still be considered meaningful.

3.2. Sample

We analyzed ScratchJr Coding Stages Assessment data of 120 K-2 teachers and 1,600 K-2 students from 43 schools across two states, collected as a part of the Coding as Another Language (CAL) project, funded by the US Department of Education (Bers et al., 2023; Kapoor et al., 2023).

3.3. Method

We analyzed the experiences of children ages 5–7 and their teachers using the app for the first time, as well as after experiencing curricular exposure. Teachers participated in professional development and children in the CAL curricular intervention for several months. Through observing performance of the target age range and those outside of the target age range (teachers), at both novice and expert time points, we measured how accessible ScratchJr is to a wide range of skill levels and how it allows users of all ages to grow in their skills when given curricular support. Furthermore, in order to explore what aspects of ScratchJr create a LFHC effect, we looked at CSA tasks with the highest success rate among a wide range of novice users, and those with the highest failure rate among experienced users. This analysis allowed us to understand which aspects of the interface and user experience lend themselves the best to creating a flexible environment at either end.

4. Results

4.1. Low floor

4.1.1. Distribution of scores

Among novice users of all ages, there was a higher proportion of lower scores than higher scores, resulting in a rightwards skew. Despite this, only a small proportion of users scored at "floor"

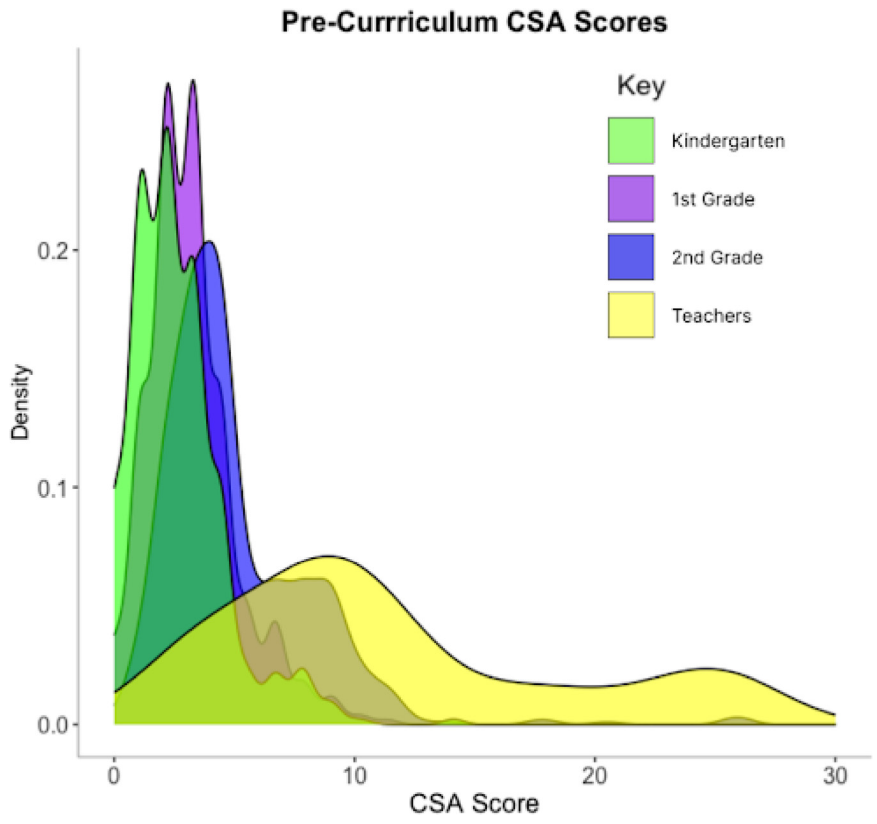


Fig. 3. CSA scores pre-curriculum.

Table 3
Floor percentages.

User Group	Percent of scores at floor (0)	Percent of scores close to floor (0 or 1 task correct)
Total	**3.71%**	**15.67%**
Kindergarten	10%	34.36%
1st Grade	3.52%	17.39%
2nd Grade	0.33%	4.47%
Teachers	0	0

(completing 0 or only 1 task correctly). Most users completed at least 1 task correctly even if they did not progress further.

Looking closely at the floor proportions across grade in Fig. 3, we see that when considering a score of 0 as the floor, only 3% of the sample is at the floor. Among Kindergarten users, this percentage grows to 10%. This indicates that the youngest users in ScratchJr's target age range are more likely to struggle when getting started.

We also examined the percent of scores *close* to the floor, meaning that the user completed one task correctly rather than none. The rationale behind this is that getting only one task correct, such as opening the app or identifying a block, still shows a low level of engagement and should be considered "floor" level. Including 1 task correct in "floor" level scores increased all percentages — Kindergarten jumped to 34%, and first grade to 17.39%. Second grade remained low at 4.47% and adults remained 0 (Table 3). This means that without curricular support, ScratchJr does have a barrier for entry for the youngest users, which does not appear with older users.

4.1.2. Item success rates

Of the interface tasks in the first stage in the CSA, the two tasks with the highest success rate among users were about (1) the block iconography at 84%, and (2) how to add a background at 82% (Table 4).

These results indicate that navigating the interface to add a background and identify a block function was understood by the majority of users. The results also show that opening the app and project editor was only accessible to 50% of the users which suggests the user flow design for initial entry to the app may be a barrier for some users. The other tasks aimed at measuring user flow for making a program, adding a character, and understanding their program, only had around 30% success rate, which means these aspects of the ScratchJr entry experience may be less intuitive.

4.2. High ceiling

In examining the distribution of scores of experienced users (students having completed the 24 lessons of the CAL-ScratchJr curriculum, and teachers completing a 4 hour of professional development), we found that the none of the age groups, including teachers, had a ceiling effect in their CSA scores, indicating that ScratchJr does have a high ceiling, even for users outside of the intended age group.

4.2.1. Distribution of scores

The distribution of children's scores following intervention (Fig. 4) remained slightly skewed right, however the scores grew

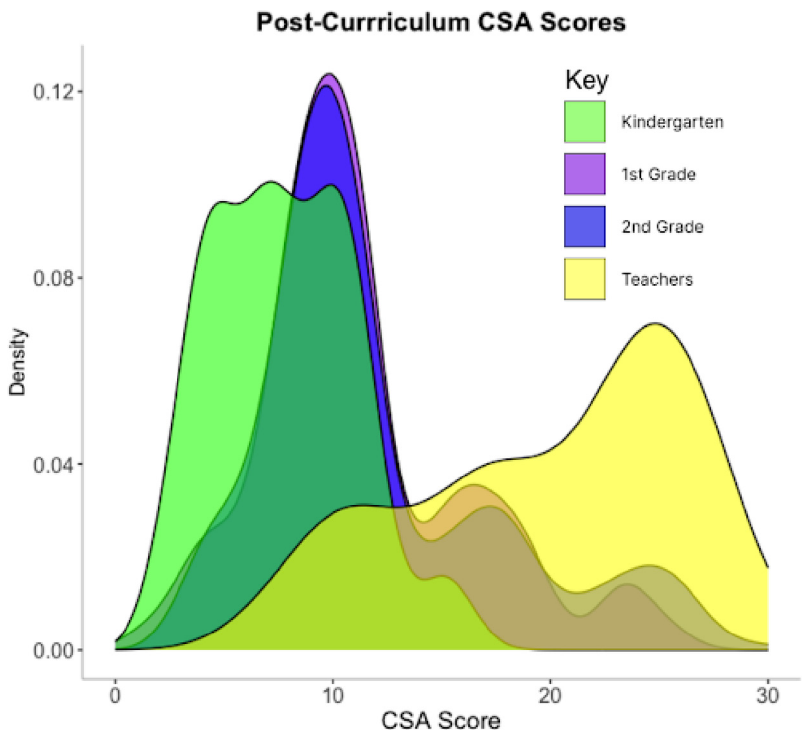


Fig. 4. CSA scores post-curriculum.

Table 4
Item success rates.

Task	What aspect of interacting with ScratchJr is being tested?	Percent correct
1.1: Please open the ScratchJr app and start a new project. *Students are assisted if unable*	• Basic knowledge of apps • Navigating home screen interface	51.06%
1.2: Can you make a program with these three blocks?	• Navigating interface — specifically selecting from the block menu and dragging blocks into the programming area • Shapes of the blocks and order matters	30.84%
1.3: Now play the program. What is Cat doing and why? *Students shown image of program in programming area if unable to complete previous question*	• Understanding their role as programmer on the app • Navigating interface to play program	33.59%
1.4: Look at this program. Which block is making Cat get bigger?	• Iconography on blocks and understanding representation	83.94%
1.5: Can you add a friend for Cat? It can be any character.	• Navigating interface - how to add multiple characters	37.94%
1.6: Can you add a background for Cat? It can be any background.	• Navigating interface - how to customize the background	81.77%

and show no floor effect, even for kindergarteners. While ScratchJr may not be accessible to 30% of the youngest users with no instruction, that percentage drops to 0% with curricular support, as shown in Table 5. No children approached the ceiling, with only 0.3% of second graders achieving the highest score on the CSA. Teachers moved to a leftwards skewed distribution, meaning there were more high scores than low, but did not approach a ceiling effect. Only 13.5% achieved the maximum score of the CSA, which means ScratchJr is a flexible enough environment for users outside of the target age range.

Table 5
Ceiling percentages.

User Group	Percent of scores at ceiling
Total	1.68%
Kindergarten	0
1st Grade	0
2nd Grade	0.36%
Teachers	13.51%

4.2.2. Item failure rates

We measured the failure rates of all tasks and found that all were fairly evenly distributed between 1%–30% failure rates. The only task that stood out was task 2.6 with a failure rate of 60%, as shown in Table 6.

Table 6
Item failure rates.

Task	What aspect of interacting with ScratchJr is being tested?	Percent incorrect
2.6. *Cat is programmed to move all the way to the Rabbit* Please program rabbit to become invisible when Cat bumps into it	• Interaction between characters • Events/Conditionals	59.8%

4.3. Comparative statistics

In addition to examining the proportions of lowest and highest scores, we also investigated whether the novice and experienced populations' difference between the lowest possible and highest possible scores respectively, are statistically significant through a one-way T-Test. The average of all pre-curriculum scores vs. 0 with a 95% confidence interval, has a p-value of <2.2e−16. Thus, the null hypothesis is rejected, confirming that the difference in the average between all novice users' scores is not equal to the floor. We also split the populations across grade (K, 1, 2, Teacher) and found that all subgroups had the same p-value of <2.2e−16. Similarly, the average of all post-curriculum scores vs. 39 (max score), with a 95% confidence interval, had a p-value of <2.2e−16. Thus, the null hypothesis is rejected, confirming that the difference in the average between all experienced users' scores is not equal to the ceiling. We also split the populations across grade (K, 1, 2, Teacher) and found that all subgroups had the same p-value of <2.2e−16. Given this analysis, we confirm that the majority of participants did not experience either a floor or ceiling effect when using ScratchJr.

5. Discussion

In this paper, by examining how novice and experienced users engaged with ScratchJr, we explored one of ScratchJr's core design principles: to be a "low floor, high ceiling" coding environment. The purpose of this work is not to show that ScratchJr has no boundaries for entry and growth, but rather to investigate where those boundaries fall along age, experience and skill levels — and whether those boundaries are appropriate for the target user group(s) of ScratchJr.

The hypothesis that ScratchJr would be accessible to entry-level children in kindergarten through second grade and their teachers, proved to be mostly supported, with the exception of kindergarten pre-curriculum performance. About 30% of kindergarteners and 17% of first graders were only able to complete one task correctly of the baseline entry tasks. This brought the entire user average to 15.7%, which, according to our determined threshold for identifying a floor, does indicate a slight floor effect in user data. This finding suggests that although ScratchJr is targeted to ages 5–7 (Kindergarten — 2nd grade in the US school system), the tool is not optimized for unsupported engagement for the youngest users and guiding activities and curriculum is needed.

There are a few possible explanations for why this floor effect occurred, the first being the users' literacy levels and how that impacts their ability to engage with the app. While ScratchJr is designed to not rely on words, but rather communicate functions through iconography, it does display the block functions as words if the block is tapped for longer than about 1 second (a long press). Many students typically discover this feature when exploring the blocks palette for the first time, however only those who are able to read gain extra support from this feature. Children in the US school system are typically learning to read throughout first grade, which could explain why the floor effect starts to diminish among first graders and is entirely gone in second grade, when it is generally expected that children have gained enough literacy skills to read single words. The other possible explanation is that children may gain significantly more technology exposure from kindergarten to second grade and have an easier time navigating the interface of any app.

Despite a floor effect occurring for the youngest users before curricular intervention, when we look at the distribution of post-curriculum scores, we see that the percentage of all users who completed no tasks correctly was 0, and those who got only one correct was just 0.002%. This finding implies the importance of introducing technological tools in early childhood with the support of adult guidance, specifically through curricular interventions which is consistent with Clarke-Midura et al.'s 2019 findings that the open-ended nature of ScratchJr may lead to a higher need for scaffolding for novice users.

The second hypothesis that ScratchJr would allow for extensive exploration and growth across experienced children ages 5–7 was fully supported by our findings: only 0.17% of users ages 5–7 reached the maximum score of the coding stages assessment, and when the population included adults, 1.7% reached the maximum score. However, reaching the maximum score of the CSA is not necessarily indicative of the true height of the ceiling in ScratchJr, because the assessment is limited to 30 items and does not exhaust all the ways ScratchJr can be used in complex, advanced, or novel ways. Therefore, we assume that the lack of ceiling effect we see in CSA scores might be even less if we were able to thoroughly measure all complex interactions afforded on ScratchJr. This further supports the notion that ScratchJr does not have a ceiling for children within the target age range of 5–7. Additionally, although we hypothesized that adults would likely reach a ceiling effect, we found that to not be the case — only 13% of adults reached the maximum score, suggesting that ScratchJr affords for growth at all ages. This is particularly noteworthy, because unlike previous flexible environments (Di Sessa, 1985; Ingalls et al., 1997; Papert, 1980; Resnick et al., 2009; Wilensky, 1999), which focused solely on lowering the floor of still robust programming languages, it was assumed that ScratchJr would have a certain degree of limitations (i.e. an "appropriately" high ceiling Flannery et al., 2013), considering how simplified it was for the target age group of 5–7. Given our findings, it could be argued that ScratchJr may be used as a learning tool outside the scope of the K-2 classroom, while still being accessible to young novice users.

In order to understand which aspects of ScratchJr were the most accessible at the novice level (lowering the floor), and the most challenging at the experienced level (raising the ceiling) we conducted an item analysis on the success and failure rates of each coding task. The high percentage of success for task 1.4, asking which block indicates the "grow" function, suggests the successful iconography of the blocks which allows all users, despite literacy levels, to engage with the app. Task 1.6's similar success rate speaks to the intuitiveness of the customization features. Task 1.6 asks the user to change backgrounds and has a success rate of 82%.

The most notable finding from item analysis was the 51% success rate among all users in the sample for task 1.1: Open the app and open a new project. This success rate dropped even lower to 32% when looking specifically at kindergarteners. Given the efforts to make a clear user flow for entering the app, this finding is surprising and indicates that the flow for entry should be more closely examined. One could argue that the task may just reflect a child's general familiarity with technology, not ScratchJr, however, a "low floor" entry point should not assume any prior experience or familiarity, especially with this age group. Therefore, we interpret this finding as an opportunity for improvement of the entry level user experience. Examples of how this could be addressed through a redesign include a short, animated tutorial showing how to open a project, and/or adding a more prominent "new project" button.

Another unexpected finding was that task 2.6, proved to be the hardest of all, with a 40% success rate (60% failure rate). The task asks the user to program one character to become invisible when another character bumps into it — with the intention of specifically probing the Start on Bump block. There are a few reasons why this task may have been difficult: (1) it is the first task to require multi-character interaction, which requires the ability to keep track of multiple programs and predict how they will interact, which requires a level of abstraction difficult at this age; (2) The low success rate could be a reflection of the Start on Bump block itself, specifically the iconography, however since we did not interview the children as to why they were unable to complete the task correctly, we are not able to make any conclusions from this finding.

Previous works (Di Sessa, 1985; Ingalls et al., 1997; Papert, 1980; Resnick et al., 2009; Wilensky, 1999) extensively explore how to lower the floor of programming environments to engage novice users of all backgrounds with complex programming. This work contributes to that body of literature by exploring how environments, specifically ScratchJr, can be designed to engage the earliest beginners: programmers in early childhood, while still offering opportunities for growth and learning. Our findings support the notion that ScratchJr is flexible enough to accommodate for the wide range of skills and abilities within its intended user age group. Finally, through a close examination of the specific user tasks, our work offers examples of how a programming environment can be optimized for early childhood through both interface and user experience design decisions.

6. Limitations and future work

The primary limitation of this study is the lack of a qualitative component to better understand the user's experience. In future studies, the ScratchJr tasks may be accompanied with a follow up interview probing the reasons behind any struggles or successes. For example, if the original task was "Please make a program using these three blocks" and the user is unable to drag the blocks into the programming area, the assessor could follow up with "Where do you think the blocks should go? Why?" This could help to better understand what aspect of the interface is acting as a barrier to engagement. That being said, the quantitative nature of this data does help to eliminate any reporting bias and allows us to examine the aggregate of thousands of users at a time. At the same time, interviewing young children is not an easy task as they might have limited abilities to explain their behaviors.

Future directions may also explore a more exhaustive method of analyzing a user's capacity for growth, especially for adults. However, because the data did not approach a ceiling effect, this concern is less crucial in the context of this investigation.

Finally, our results suggest that ScratchJr affords for knowledge growth for all users, not just children between the ages of 5–7. Future studies should investigate how ScratchJr may be utilized as an educational tool for users outside of the intended age range — such as older children or adults.

7. Conclusion

In this paper we investigated the bounds of the "low floor, high ceiling" design of ScratchJr. To do this, we looked at both novice and experienced users, and both children ages 5–7 and their teachers. We found that the youngest, and most novice users were all able to engage with the app to some extent with no instruction, but engagement was optimized with curricular support. Additionally, we found that the oldest, and most experienced users still had room for growth and discovery in the app, showing the flexible nature of ScratchJr. We conclude that the design decisions such as iconography, block variation, and open-endedness of the environment creates a welcoming and engaging experience for a wide range of users, both children and adults.

Selection and participation and consent statement

All the study's participants were students and teachers from public schools across two states in New England as a part of the Coding as Another Language ScratchJr Randomized Control Trial. Informed parental consent for all participants, as well as informed assent for all participants over age 7, was obtained.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

We would like to express our gratitude to the teachers, administrators, and children who participated in this study. Their time, effort, and cooperation were essential to the success of our research.

Ethical statement

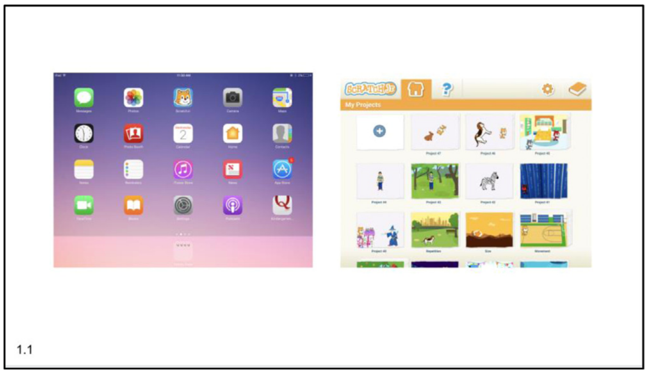
All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional research committee and with the 1964 Helsinki Declaration and its later amendments or comparable ethical standards.

Funding

This research was generously supported by the US Department of Education.

Appendix A

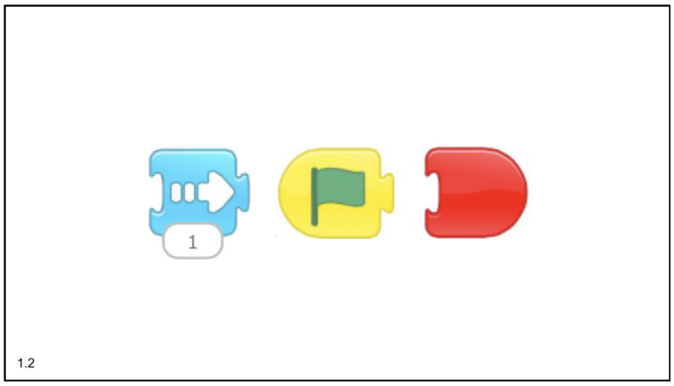
Task 1.1 of Coding Stages Assessment


1.1

Note. The assessor asks: "Can you open the ScratchJr app and open a new project?"

Appendix B

Task 1.2 of Coding Stages Assessment

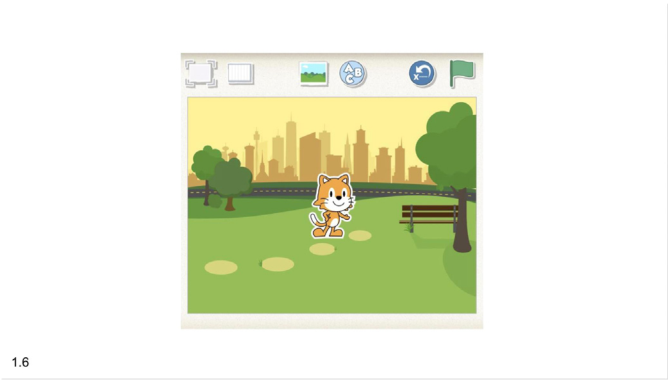


1.2

Note. The assessor asks: "Can you make a program using these three blocks?"

Appendix C

Task 1.6 of Coding Stages Assessment


1.6

Note. The assessor asks: "Can you change the background?"

References

Ben-Ari, M. (1998). Constructivism in computer science education. In *Proceedings of the twenty-ninth SIGCSE technical symposium on computer science education - SIGCSE '98, vol. 30, no. 1* (pp. 257–261). http://dx.doi.org/10.1145/273133.274308.

Berland, M., Martin, T., Benton, T., Smith, C., & Davis, D. (2013). Using learning analytics to understand the learning pathways of novice programmers. *The Journal of the Learning Sciences*, *22*(4), 564–599, http://www.jstor.org/stable/43828324.

Bers, M. (2019). Coding as another language: a pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education*, *6*(4), 499–528. http://dx.doi.org/10.1007/s40692-019-00147-3.

Bers, M. (2020). *Coding as a playground: Programming and computational thinking in the early childhood classroom* (2nd ed.). New York, NY: Routledge Press.

Bers, M., Blake-West, J., Kapoor, M., Levinson, T., Relkin, E., Unahalekhaka, A., & Yang, Z. (2023). Coding as another language: Research-based curriculum for early childhood computer science. *Early Childhood Research Quarterly*, *64*, 394–404. http://dx.doi.org/10.1016/j.ecresq.2023.05.002.

Bers, M., Doyle-Lynch, A., & Chau, C. (2012). Positive technological development. In *Constructing the Self in a Digital World* (pp. 110–136). http://dx.doi.org/10.1017/cbo9781139027656.007.

Bers, M., Flannery, L., Kazakoff, E., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*, 145–157.

Bers, M., & Resnick, M. (2015). *The official ScratchJr book: Help your kids learn to code!*. No Starch Press.

Bers, M., & Sullivan, A. (2019). Computer science education in early childhood: The case of scratchjr. *Journal of Information Technology Education: Innovations in Practice*, *18*, 113–138.

Betancur, L., Votruba-Drzal, E., & Schunn, C. (2018). Socioeconomic gaps in science achievement. *International Journal of STEM Education*, *5*(1), http://dx.doi.org/10.1186/s40594-018-0132-5.

Blake-West, J., & Bers, M. U. (2023). *ScratchJr Connect: Sharing resources for digital making around the world [Poster session]*. Montréal, QC, Canada: International Society of the Learning Sciences, https://sites.bc.edu/devtech/wp-content/uploads/sites/113/2023/06/ScratchJrConnect_ISLS2023_FinalJBW.pdf.

Clarke-Midura, J., Lee, V., Shumway, J., & Hamilton, M. (2019). The building blocks of coding: A comparison of early childhood coding toys. *Information and Learning Science*, *120*(7/8), 505–518. http://dx.doi.org/10.1108/ILS-06-2019-0059.

Code. org (2018). *2018 annual report*. Code.org., Retrieved from https://code.org/files/annual-report-2018.pdf.

de Ruiter, L., & Bers, M. (2021). The Coding Stages Assessment: Development and validation of an instrument for assessing young children's proficiency in the SCRATCHJR programming language. *Computer Science Education*, *32*(4), 388–417. http://dx.doi.org/10.1080/08993408.2021.1956216.

Di Sessa, A. (1985). A principled design for an integrated computational environment. *Human-Computer Interaction*, *1*(1), 1–47.

Erikson Institute (2016). Technology and young children in the digital age: A report from the Erikson Institute. https://www.erikson.edu/wp-content/uploads/2018/07/Erikson-Institute-Technology-and-Young-Children-Survey.pdf.

Fayer, S., Lacey, A., & Watson, A. (2017). *BLS spotlight on statistics: STEM occupations-past, present, and future*. Washington, DC: U.S. Department of Labor, Bureau of Labor Statistics.

Flannery, L., Silverman, B., Kazakoff, E., Bers, M., Bontá, P., & Resnick, M. (2013). Designing ScratchJr. In *Proceedings of the 12th international conference on interaction design and children*. http://dx.doi.org/10.1145/2485760.2485785.

Guzdial, M. (2004). Programming environments for novices. *Comput. Sci. Edu. Res.*, 127–154.

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

Ingalls, D., Kaehler, T., Maloney, J., Wallace, S., & Kay, A. (1997). Back to the future: The story of Squeak, a practical Smalltalk written in itself. *SIGPLAN Notices*, *32*(10), 318–326. http://dx.doi.org/10.1145/263700.263754.

Kapoor, M., Yang, Z., & M., B. (2023). Supporting early elementary teachers' coding knowledge and self-efficacy through virtual professional development. *Journal of Technology and Teacher Education*, *30*(4), 1–31, 2023.

Kelleher, C., & Pausch, R. (2003). *Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers, vol. 37* (2nd ed.). (pp. 83–137). Carnegie Mellon University.

Maloney, J., Peppier, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: Urban youth programming learning with scratch. In *Proceedings of the 39th SIGCSE technical symposium on computer science education* (pp. 367–371). New York, NY: ACM, http://dx.doi.org/10.1145/1352135.1352260.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education*, *10*(4), 1–15. http://dx.doi.org/10.1145/1868358.1868363.

McHorney, C., & Tarlov, A. (1995). Individual-patient monitoring in clinical practice: Are available health status surveys adequate? *Quality of Life Research*, *4*(4), 293–307. http://dx.doi.org/10.1007/bf01593882.

McKeithen, K., Reitman, J., Rueter, H., & Hirtle, S. (1981). Knowledge organization and skill differences in computer programmers. *Cognitive Psychology*, *13*(3), 307–325. http://dx.doi.org/10.1016/0010-0285(81)90012-8.

Medhi, I., Patnaik, S., Brunskill, E., Gautama, S., Thies, W., & Toyama, K. (2011). Designing mobile interfaces for novice and low-literacy users. *ACM Transactions on Computer-Human Interaction*, *18*(1), 1–28.

Myers, B., & Stylos, J. (2016). Improving API usability. *Communications of the ACM*, *59*, 62–69. http://dx.doi.org/10.1145/2896587.

National Center for Science Engineering Statistics (NCSES) (2023). *Diversity and STEM: Women, minorities, and persons with disabilities 2023*: *Special report NSF 23-315*, Alexandria, VA: National Science Foundation, Available at https://ncses.nsf.gov/wmpd.

Norman, D., & Draper, S. (1986). *User centered system design: New perspectives on human-computer interaction*.

NRICH Team (2019). *Creating a low threshold high ceiling classroom*. University of Cambridge: Faculty of Mathematics, Retrieved March 20, 2023, from https://nrich.maths.org/7701.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.

Portelance, D., Strawhacker, A., & Bers, M. (2015). Constructing the ScratchJr programming language in the early childhood classroom. *International Journal of Technology and Design Education*, 1–16. http://dx.doi.org/10.1007/s10798-015-9325-0.

Rader, C., Brand, C., & Lewis, C. (1997). Degrees of comprehension. In *Proceedings of the ACM SIGCHI conference on human factors in computing systems* (pp. 351–358). http://dx.doi.org/10.1145/258549.258793.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137–172.

Shertz, J., & Weiser, M. (1981). A study of programming problem representation in novice and expert programmers. *Proceedings of the eighteenth annual computer personnel research conference* (pp. 302–322).

Soloway, E. (1986). Learning to program= learning to construct mechanisms and explanations. *Communications of the ACM*, *29*(9), 850–858.

Strawhacker, A., Lee, M., Caine, C., & Bers, M. (2015). ScratchJr Demo: A coding language for Kindergarten. In *Proceedings of the 14th international conference on interaction design and children*. New York, NY: ACM.

Sullivan, A., & Bers, M. (2016). Girls. *Boys, and bots: Gender differences in young children's performance on robotics and programming tasks*, *15*, 145–165, Retrieved from http://www.informingscience.org/Publications/3547.

Vygotsky, L. (1978). *Mind in society: Development of higher psychological processes*. Harvard University Press.

Wiedenbeck, S. (1985). Novice/expert differences in programming skills. *International Journal of Man-Machine Studies*, *23*(4), 383–390.

Wilensky, U. (1999). *NetLogo [Computer software]*. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Retrieved from http://ccl.northwestern.edu/netlogo.