StoryCoder: Teaching Computational Thinking Concepts Through Storytelling in a Voice-Guided App for Children

Griffin Dietz
dietz@stanford.edu
Computer Science Department
Stanford University
Stanford, CA, USA

Jimmy K. Le
jimmyle@stanford.edu
Computer Science Department
Stanford University
Stanford, CA, USA

Nadin Tamer
nadint@stanford.edu
Computer Science Department
Stanford University
Stanford, CA, USA

Jenny Han
jlhan@stanford.edu
Computer Science Department
Stanford University
Stanford, CA, USA

Hyowon Gweon
hyo@stanford.edu
Department of Pyschology
Stanford University
Stanford, CA, USA

Elizabeth L. Murnane
emurnane@dartmouth.edu
Thayer School of Engineering
Dartmouth College
Hanover, NH, USA

James A. Landay
landay@stanford.edu
Computer Science Department
Stanford University
Stanford, CA, USA

ABSTRACT

Computational thinking (CT) education reaches only a fraction of young children, in part because CT learning tools often require expensive hardware or fluent literacy. Informed by needfinding interviews, we developed a voice-guided smartphone application leveraging storytelling as a creative activity by which to teach CT concepts to 5- to 8-year-old children. The app includes two storytelling games where users create and listen to stories as well as four CT games where users then modify those stories to learn about sequences, loops, events, and variables. We improved upon the app design through wizard-of-oz testing ($N = 28$) and iterative design testing ($N = 22$) before conducting an evaluation study ($N = 22$). Children were successfully able to navigate the app, effectively learn about the target computing concepts, and, after using the app, children demonstrated above-chance performance on a near transfer CT concept recognition task.

CCS CONCEPTS

• **Social and professional topics → Computational thinking**; **Children**; • **Human-centered computing →** *Natural language interfaces.*

KEYWORDS

children, computational thinking, storytelling, voice user interface

ACM Reference Format:
Griffin Dietz, Jimmy K. Le, Nadin Tamer, Jenny Han, Hyowon Gweon, Elizabeth L. Murnane, and James A. Landay. 2021. StoryCoder: Teaching Computational Thinking Concepts Through Storytelling in a Voice-Guided App for Children. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3411764.3445039

1 INTRODUCTION

In recent years, there has been a global push for computing education for all. This trend is largely driven by the need for people to understand our increasingly technological world, a shortage of diverse qualified employees for the technical workforce, and a growing necessity for computing skills across a broad range of jobs. Critically, there are benefits to computing education even in early childhood: early computer science (CS) exposure increases later interest in the field among female and minority students [34, 35], contributes to the development of computational thinking (CT) and computational literacy [9], and builds lifelong skills and readiness for a child's educational career [39].

Despite the benefits of such early exposure to computer science and evidence suggesting children can cognitively engage with CT concepts [20], the available infrastructure for teaching computing to early elementary age (K–2) children does not meet all students' needs. In particular, children often lack the advanced literacy, numeracy, and fine motor skills needed to use most traditional programming environments or CT tools. Existing solutions for teaching computing to children are often unable to remove this literacy threshold [67], depend on a knowledgeable teacher for students to make meaningful progress [54, 67], or require expensive or specialized technologies that are unaffordable for many school districts and not readily available to all students [42, 79].

However, if we can teach CT in a manner synergistic with promoting reading readiness, literacy will no longer be a barrier to early

CS education. Here, we argue that children can learn CT concepts in a way that *simultaneously builds literacy skills* by engaging with these ideas through storytelling. Storytelling in early childhood can enhance language and literacy development and contribute to improved oracy, listening, reading, and writing skills later in life [55, 62]. Furthermore, much of everyday interaction centers on sharing experiences through storytelling [56], and only through play and practice do children develop the tools for effective social communication [59].

By leveraging storytelling as a domain to teach computational thinking, children can build critical foundational skills in these two areas, which are a natural fit with one another. Children's stories, like code, are told in logical sequences, often leverage repetition (i.e., loops), and have components (e.g., characters and locations) that can be changed, reused, or replaced (i.e., data or variables). Stories are also built on top of abstractions (e.g., a standard story structure) and are logically organized using decomposition (e.g., scenes or chapters). Thus, storytelling presents an opportunity for teaching key computing concepts in a way that promotes creativity, supports self-expression, and simultaneously builds children's computing and literacy skills.

Aligning with the oral traditions of storytelling, it is possible to introduce these computing concepts using a voice-based interface, which can also serve to alleviate the literacy requirements present in most existing solutions. By designing a voice interface for informal, at-home learning that runs on accessible hardware (smartphones), we can promote a style of computing education that reaches a far greater number of children. This approach will facilitate access to CS education, especially for those who do not have such materials available in their schools or whose families are unable to purchase specialized hardware for use at home.

Building on related work and our own formative investigation, we introduce StoryCoder, a voice-based smartphone application for early school-aged children (ages 5–8) that introduces coding concepts through storytelling activities. This system supports children in the creation of their own stories by teaching the conventional story structure introduced in many early elementary classrooms. It then allows children to use and modify those stories through concept-targeted story games to learn about and engage with four key ideas in computing: sequences, loops, events, and variables.

In this paper we present the following main contributions:

- A formative investigation of educator and student needs that suggests computational thinking tools should run on accessible hardware, not require literacy skills, and leverage interdisciplinary, personal, and creative activities that can combat student self-doubt and increase engagement.
- StoryCoder, a voice-based smartphone application that introduces children to computing concepts through storytelling activities.
- A multi-day user study evaluating the computational and storytelling learning potential of StoryCoder and children's attitudinal perceptions toward the system and computing more broadly.

Overall, our evaluation demonstrates that StoryCoder—and the storytelling-based, voice-guided approach it instantiates—effectively introduces target computing concepts and engages children in an activity they perceive as helpful for later success in a formal computer class. Furthermore, the system provides measurable benefits, with children demonstrating above-chance performance on a near transfer post-assessment recognition task and with older children creating better stories than they do without the system, as evaluated by a standard rubric for narrative assessment.

2 RELATED WORK

Here we describe related work on curriculum and learning technologies for early computing and storytelling education, and on voice interfaces for preliterate children.

2.1 Computational Thinking in K–2

2.1.1 Computational Thinking Concepts and Curriculum. Since Jeanette Wing's influential article was published in 2006 [83], computational thinking (CT)—a term first coined by Seymour Papert in 1980 [60]—has received considerable attention. While the precise definition and makeup of this computational problem-solving skill set are still under debate, the literature generally agrees on the importance of practices such as decomposition and modularity and, to some degree, on specific computational concepts such as parallelism and conditional reasoning [6, 17, 38, 81, 83, 84]. Brennan and Resnick's computational thinking framework—which is rooted in observations of Scratch [67] users and is the only CT conceptualization formulated to-date with primary-school aged learners specifically in mind—calls out specific computing concepts (e.g., sequences and loops), practices (e.g., abstracting and modularizing), and perspectives (e.g., expressing and connecting) that are core to computational thinking in early education [12].

Recently, national efforts in the United States have led to curricular frameworks for computer science education [4, 18], which delineate specific grade-level learning objectives. Cross-referencing these frameworks with Brennan and Resnick's CT definition [12], we identify an intersection of target concepts for early school age children: sequences, loops, events, and data/variables as *computational concepts*; abstraction, planning, and modularity as *computational practices*; and expressing oneself and connecting with others as *computational perspectives*. Our research and design work therefore specifically focuses on teaching these CT concepts, while providing built-in scaffolding to support these practices and perspectives.

2.1.2 Computational Thinking Learning Technologies. In recent decades, a number of tools have been created to teach CS and CT to children. Many utilize new languages or games developed specifically to teach programming to young audiences [19, 45, 50, 60, 61]. However, literacy is a prerequisite to their usage, creating a barrier of entry for many of the youngest learners. To make computing more accessible to this preliterate demographic, researchers have introduced block-based programming, programmable robots, and unplugged activities.

Block-based languages disguise the underlying programming syntax using blocks that fit together only when syntactically correct. However, Scratch [67] and Blockly [30]—arguably the most prominent block-based languages—still incorporate written text, and therefore can only effectively reach older users. To reach a younger audience, other languages following this paradigm have replaced text with symbols, thereby removing the need to know how to read and write [27, 44]. This change allows children to create programs by piecing visual—and sometimes tangible [42]—blocks together to represent different coding structures. However, these languages often require a more experienced teacher to encourage best practices or to guide the student toward building more complex programs [54].

Programmable robots present another common paradigm for introducing computational thinking in early education. These robots support children in learning about computing through physical play by mapping programmatic commands to actions in the physical world. While many such robots still leverage block-based programming as a means to give instructions to the robot (e.g., KIBO [76] and Dash Robot [85]), some target the teaching of CT concepts or practices without specific programming languages. The Bee-Bot, for example, teaches sequencing to children in kindergarten through second grade using small robots with built-in directional command buttons [79]. While these programmable robots are engaging and effective, this technology is often expensive, and purchasing a full set for a classroom (or even one for home use) is financially infeasible for many schools or parents.

Given the scarcity of access to such digital tools and resources, many educators have developed or leveraged CS Unplugged [8] activities that allow children to engage with computational concepts without any technology. For instance, Robot Turtles [72] is a board game for children as young as three that introduces basic programming concepts, and Hello Ruby [51] is a children's book series that teaches about computers, technology, and programming. However, while these "unplugged" activities are relatively accessible and approachable ways to introduce children to CS ideas, research suggests that students may not connect this style of learning back to computing [78].

In summary, there are three dominant paradigms in early computing education: block-based programming, programmable robots, and unplugged activities. While each of these approaches have their merits, computing education still fails to reach a large proportion of the young population due to literacy requirements, a shortage of educators, expensive hardware, and/or a disconnect between materials and objectives. These drawbacks are our key motivation behind investigating additional paradigms that we can leverage for early childhood computing education.

2.2 Storytelling in K–2

2.2.1 Storytelling Curriculum. While the push for CT education is relatively new, there has been a longstanding interest and effort to teach storytelling to children. Storytelling is a vital lifelong communication skill that fosters the growth of relationships [55], and research has shown that language and listening comprehension, built through exposure to storytelling, are critical to academic success [55, 62]. In our effort to teach storytelling to support the acquisition of CS and CT concepts, we also look to storytelling curriculum and technologies as a source of inspiration.

Reading aloud is one common educational activity and presents a valuable way to promote literacy growth in emergent readers [1, 25]. Such practice teaches children about the difference between spoken and written language, builds an understanding that the written word is a representation of speech, and may contribute to letter or word recognition [25]. Reading aloud also exposes children to story structure (e.g., stories have a beginning, middle, and end), which is critical to understanding, and later constructing, written texts [25].

In fact, research has shown that explicitly teaching story structure to children increases their language comprehension skills, improving both their story memory and comprehension [7, 32]. Children recall more concepts from new stories and answered more questions about the structural elements of such stories after receiving explicit instruction on story structure, as compared to children without such instruction [75]. Therefore, it is unsurprising that much of early literacy education focuses on reading, listening to, and understanding stories.

2.2.2 Technologies for Supporting Storytelling. Researchers have developed a number of technologies intended to directly support young children's storytelling (e.g., [11, 28, 40, 52]), which range from audio-supported physical experiences to fully digital experiences. Many existing systems mix physical and digital formats by allowing children to record and playback stories surrounding a limited set of tangible props [13, 15, 33, 74]. Children use these props as tangible manipulatives that represent specific story elements, leading to creative and collaborative physical play. However, some criticize these systems because compatible props may constrain the expanse of creativity. TellTable resolves this constraint by allowing students to create stories on a large multi-touch surface with support for incorporating any physical object into the story [14]. It elicits the creation of stories, allows children to take inspiration from other stories, and fosters the development of a community of story creators. On the other end of the physical-digital spectrum, there are a collection of phone- and tablet-based systems (e.g., Fiabot [68] and StoryBank [31]) that support multimedia story creation by allowing users to photograph drawings or surroundings to incorporate into their tales. By using more readily available devices, these tools can reach a broader range of children from diverse socioeconomic backgrounds.

Additional motivations for children's storytelling include improving children's health, building competencies in other academic subjects, or supporting personal relationships. Indeed, there have been several devices that support children in therapy [64], teach math [2] or foreign languages [89], or allow for asynchronous or remote storytelling as a means to connect distanced family members (e.g., grandparents) [40, 66, 80].

Many of the aforementioned storytelling systems were designed to foster children's creativity, communication, and fantasy play. However, while they demonstrate the capacity for technology to support storytelling in young users, they do not explicitly teach formal story structure to children. Toontastic is a notable exception [69]. Originally designed for a large display with multi-pen input, it has since been adapted and commercially released for smartphones, and scaffolds the storytelling process by breaking it into parts [69]. In doing so, it explicitly encourages a child to consider a beginning-middle-end structure for their stories. However, there has been no formal investigation into the learning outcomes of this system [69], or systems that introduce story structure more broadly.

2.3 Technologies That Simultaneously Support Computing and Storytelling

The creative potential, engagement opportunities, and underlying structure of stories all motivate storytelling as a promising domain for introducing computing concepts to children. People are driven to see their ideas realized in the real world [71], and prior work demonstrates that interdisciplinary approaches that teach STEM disciplines through creative activities, like storytelling, engage students by allowing them to make projects that are personally meaningful [49].

Indeed, several CS learning tools geared at older children have built-in support for visual storytelling, particularly via animation [19, 26, 46, 67], while other systems support programmable characters [70], storytellers [24], and listeners [10]. CyberPLAYce expands this storytelling further into the physical world, by allowing upper-elementary age children (8–12 years old) to physically recreate a story's setting using electronic modules, thereby supporting both storytelling and CT practices [73]. Solely audio-based programming tools that leverage storytelling have been used to support accessibility in computing education for visually impaired users, but they are not directly geared at preliterate children [48]. There are no prior systems that specifically target the teaching of computing concepts through storytelling activities to pre- and early-readers as a means to simultaneously build both early computing and literacy skills.

2.4 Voice Interfaces for Preliterate Children

Voice interfaces can remove the literacy barriers of many programming environments, while satisfying the need for children to listen to stories [1] and hear their own voice in their creations [15]. As conversational agents grow increasingly advanced, children have begun to see them as intelligent and friendly [23], and with appropriate scaffolding, children can demonstrably learn from their interactions with these systems [86]. With voice interfaces' rapid gains in popularity and performance [57, 87], such technology presents a promising modality for educating young children.

To date, many commercial voice-based apps exist that tell stories to children (e.g., Amazon Storytime), add pre-established sound effects to a fixed list of children's books (e.g., Disney Read Along), present choose-your-own-adventure stories (e.g., The Magic Door), play mad-lib style games (e.g., Story Blanks), and more. However, none simultaneously supports storytelling practice and the learning of computational concepts.

3 FORMATIVE INVESTIGATION

As detailed above, there are already an array of systems, tools, and approaches aimed at supporting early childhood computing education. Yet such learning opportunities are still reaching only a fraction of children [36]. We engaged in a formative investigation to 1) understand the disconnect between existing solutions and educator and student needs and 2) inform design decisions that target the correct challenges. The Stanford IRB approved all procedures.

3.1 Method

We interviewed seven elementary school computing educators (1 female, 6 male) and four child programmers (ages 6–12; 3 female, 1 male). To be inclusive of diverse experiences, we recruited educators who work in a variety of settings, including formal classrooms, curriculum development offices, and after-school programs, as well as children who have a mix of formal, informal, and at-home computing instruction. The interviews followed a semi-structured format and included questions relating to experiences teaching or learning computer science, typical learning activities, the challenges and motivations for such teaching and learning, and specific questions about what these participants felt was missing from their current practice. The interviews lasted 34 to 63 minutes ($M = 48.14$) for educators and 10 to 29 minutes ($M = 19.25$) for students. Throughout the interview, the interviewer took notes on participant responses and asked targeted follow-up questions, and audio recordings of the sessions were transcribed afterwards for later analysis.

In addition, we conducted two classroom observations in informal education settings. One observation was in a 45-minute Scratch-based class for 4–9 year old students in a program that provides free CS classes to underrepresented and low-income students. The other was in a 2-hour paid after-school program teaching Python to 9–12 year old students. During these observations, the researcher sat in the back of the classroom and took notes on classroom behaviors, class activities, and notable student and instructor interactions. During both observational sessions, students approached the researchers for help on their programs. In these situations, as agreed upon with the instructor in advance, the researcher assisted the student and then recorded targeted observational notes about this direct interaction.

3.2 Key Findings & Design Implications

From this process, we identified three high-level findings with corresponding design implications reflecting both the challenges that educators and learners face as well as the approaches they implement to circumvent those challenges.

3.2.1 Accessibility. For many students, we identified that computing education is not accessible because they do not have relevant curriculum in their schools and may not have computers or specialized hardware at home. In addition, we found that even when children have access to such technologies or instruction, they still lack the fine motor skills needed to succeed with traditional input mechanisms (e.g., mouse and keyboard). Finally, it is important to remember that our youngest students are still building foundations in literacy and may not reliably be able to read and write; indeed, we saw that even if children can access CS learning technologies, they may not have the literacy skills to use them, which can also exacerbate computational literacy challenges if navigation relies on textual labels. Altogether, to address challenges of access we aim to build a system that 1) targets at-home use so children without formal computing curriculum still have an opportunity to engage with the material, 2) uses widely accessible, rather than specialized, hardware, and 3) is completely navigable without any literacy skills.

3.2.2 Approachability. Next, our observations revealed that even if students can access computing education tools, educators and parents may have self-doubt in their ability to support students, and students may not see themselves as someone who can successfully learn about CS. This self-doubt can lead to reticence or avoidance, but, critically, students often do not develop this doubt until they get older [37]. Early exposure to CS means that children can engage with these ideas while they are still developing their identity, thereby building technology proficiency into their sense of self. Expanding on prior work and on approaches to combat self-doubt that we heard about in our interviews, we aim to 1) leverage interdisciplinary topics to convey CS concepts [49] and 2) make learning personally meaningful by connecting it to learner interests [41].

3.2.3 Engagement. Finally, as repeatedly mentioned by educators and observed in classrooms, young children have particularly *short* attention spans; yet for them to learn, we need to drive more *frequent, long-term* usage. We saw that the most immediate goal of educators (outside direct content teaching) is to engage their students, and students themselves want to use engaging tools. Educators and students alike described a need to dive right into the material by providing just-in-time support rather than extensive up-front instructions. In addition, they discussed how incorporating creative activities, like drawing, dance, or music can help to increase interest, along with the ability to jointly engage with friends or family, which research shows can increase learning outcomes [77]. Based on these comments, we aim to promote engagement by 1) providing scaffolding in the moment, 2) letting children be creative while learning, and 3) supporting collaboration and co-play.

4 SYSTEM DESIGN AND DEVELOPMENT

Based on the above design insights we built StoryCoder, a smartphone-based voice interface to introduce computational concepts to children (aged 5–8) through storytelling activities.

4.1 Design Objectives

In line with our previously stated accessibility design goals, smartphones are the most widely accessible hardware, with 94% of children in the U.S. having internet access at home, but 6% of those (primarily low-income minorities) only via smartphones [29]. Voice interfaces alleviate literacy requirements, and a conversational system can guide children in-the-moment using appropriate conversational fallbacks. We therefore designed our app to run on smartphones and be a voice-first interaction, although we do additionally allow for multi-touch input to reduce memory load in more complex selection tasks.

To support our approachability and engagement goals, we leverage storytelling as an interdisciplinary activity that simultaneously builds linguistic and communicative skills while teaching computational thinking concepts. This storytelling approach provides a creative outlet for children to share their own thoughts and ideas while engaging with this educational content, alone or with a collaborator.

From a CT education perspective, we aimed to directly introduce the concepts of sequences, loops, events, and variables, while scaffolding the practices of abstraction, planning, and modularity and the perspective of expressing oneself. To support storytelling, we provide clear guidance on story structures and reflective listening comprehension activities, while still allowing children to tell open-ended stories.

We also continuously designed for active, engaged, meaningful, and socially interactive learning to strengthen the educational validity of our system [41]. That is, we ensured children were thinking critically about the questions presented and participating in minds-on learning by including purposeful interactions (e.g., reflection questions) surrounding abstract concepts. We supported engagement through contingent interactions, relevant feedback, and process praise, rather than distracting content. We built meaning into children's learning by allowing them to draw connections between new material and topics of personal interest via the stories they create. Finally, we embedded this experience in a socially interactive and contingently responsive system that supports both co-play and virtual interactions.

4.2 User Flows

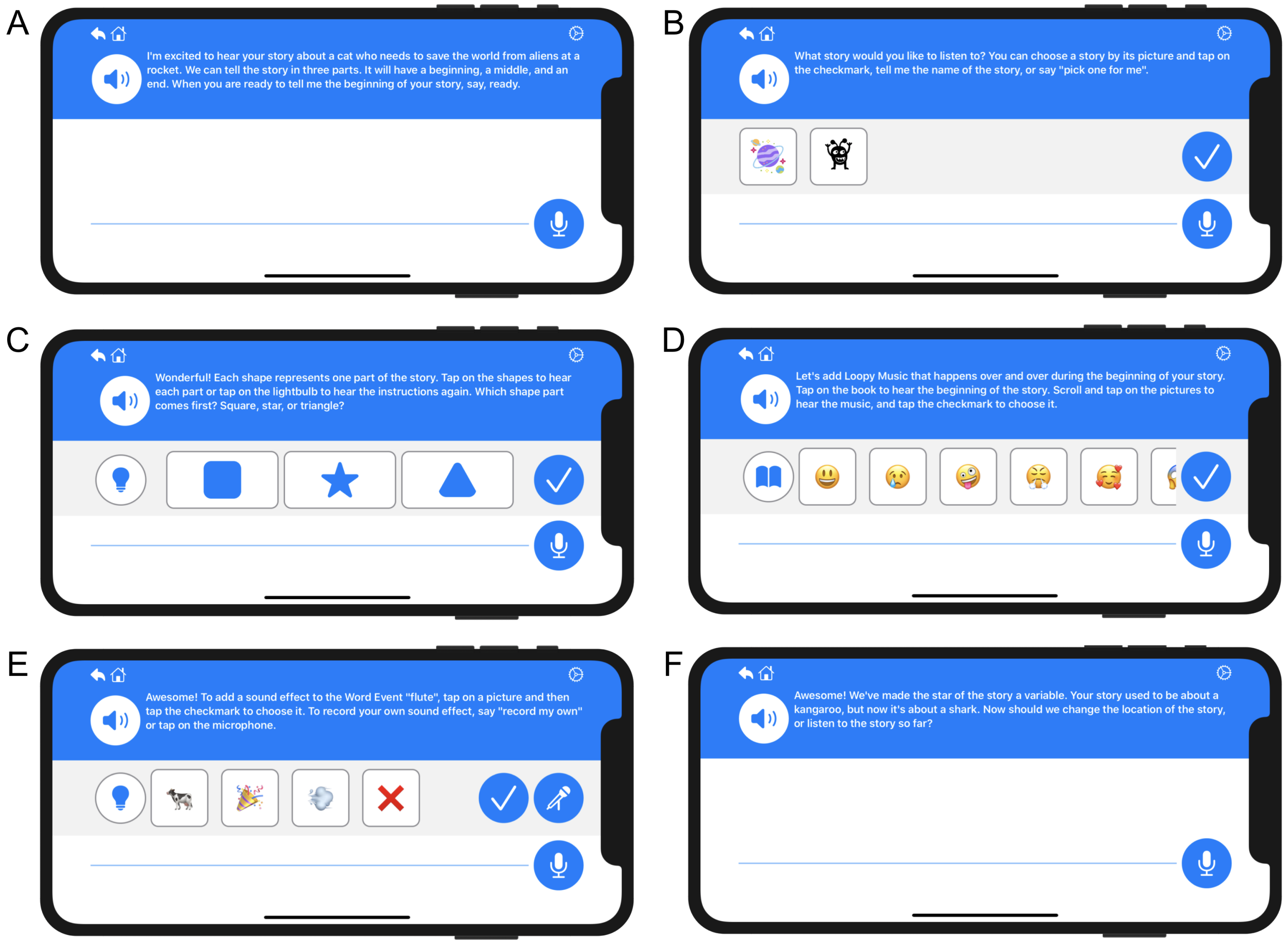
Our app consists of six voice-navigable user flows: two targeting storytelling that guide children in creating and listening to their stories and four "silly story games" that each target a specific computational concept (see Figure 1).

4.2.1 Create a Story. The Create a Story flow, central to the entire app, aims to directly introduce both planning and story structure to children while indirectly presenting this story structure as an abstraction generalizable across stories. Specifically, StoryCoder aids children's story planning by asking a series of probing questions about the main character, setting, and problem to help children create a story plan. The app then builds scaffolds around formal story structure by guiding the user in telling the story in three distinct parts—a beginning, a middle, and an end—while suggesting what content should go in each part (e.g., *"In the beginning we introduce the star and the location of the story"*). Children are prompted to think about their story until they are ready to tell it, and once they begin, the microphone remains open until the child signals they are done telling that part. Upon finishing, the child has the option to listen to their story told back to them and to save it to their story database for future listening.

4.2.2 Listen to Stories. The Listen flow allows children to visit their story database to listen to the stories they have previously told or to built-in stories that come with the app. This flow targets listening comprehension skills by reading stories aloud to children and asking them a follow-up reflection question about a core structural element (e.g., *"What was the problem in that story?"*).

4.2.3 Sequences. The sequences game, called Scrambled Sequences, is the first of four "silly story games." In this game, we introduce the idea that sequences are "the order that different steps should happen in" by scrambling parts of the story and asking children to put them back into the correct sequence. Whether correct or not, children can listen to the order they've chosen as a way to understand the effects of reordering and the idea of a correct order.

4.2.4 Loops. The loops game, Loopy Music, teaches that a loop is "something that happens over and over again." In this game, children select short music clips that repeatedly loop as background music during each part of their story. Children can listen to each story part before making a music selection (with the musical mood/tone



I'm excited to hear your story about a cat who needs to save the world from aliens at a rocket. We can tell the story in three parts. It will have a beginning, a middle, and an end. When you are ready to tell me the beginning of your story, say, ready.

What story would you like to listen to? You can choose a story by its picture and tap on the checkmark, tell me the name of the story, or say "pick one for me".

Wonderful! Each shape represents one part of the story. Tap on the shapes to hear each part or tap on the lightbulb to hear the instructions again. Which shape part comes first? Square, star, or triangle?

Let's add Loopy Music that happens over and over during the beginning of your story. Tap on the book to hear the beginning of the story. Scroll and tap on the pictures to hear the music, and tap the checkmark to choose it.

Awesome! To add a sound effect to the Word Event "flute", tap on a picture and then tap the checkmark to choose it. To record your own sound effect, say "record my own" or tap on the microphone.

Awesome! We've made the star of the story a variable. Your story used to be about a kangaroo, but now it's about a shark. Now should we change the location of the story, or listen to the story so far?

Figure 1: Screenshots of the six primary user flows in the StoryCoder app: A) Create a Story, B) Listen to Stories, C) Scrambled Sequences, D) Loopy Music, E) Word Events, and F) Very Variable.

represented by an emoji). After selecting music for all three parts, users can listen to their story with the musical accompaniment.

4.2.5 Events. The events game, called Word Events, introduces events as "something that happens that causes something else to happen." In this game, children select words in their story as events and then choose or record their own sound effects to play when these words are read. For example children might make the word 'cow' into an event, so whenever it is read, it triggers a sound effect, playing a recorded 'moo' sound. Users can listen to their story after adding each sound or at the end of adding all their sounds.

4.2.6 Variables. Finally, the variables game, called Very Variable, teaches that variables are "things that can change, but variables can only change for another thing of the same type." In this game, the child can pick certain words in their story (i.e., main character, location, nouns, or adjectives) to change for other words of the same type (e.g., change a forest location to a castle location). The system will replace these words in the story for the newly chosen word while keeping the remainder of the story the same. After each change or after making all their changes, the child can listen to the newly modified story.

4.3 Design Process

The creation of each user flow followed the same five step voice-first design process:

4.3.1 Brainstorming. For each flow, we began with a team brainstorm, each involving 3–5 researchers. These sessions would begin with a high level introduction of the target CT concepts. The team would then ideate ways to apply these concepts to creative activities (e.g., deciding to connect computing and storytelling during one of our earliest brainstorms) and then to specific story games thereafter (e.g., teaching events through triggered sound effects). We next narrowed down these ideas and identified the most promising ones to turn into conversation flow prototypes.

4.3.2 *Conversation Flows.*

For each top idea generated during brainstorming, we created a conversation flow. These conversation flows resemble flow charts, where every node is something the system would say and every connection is a potential response from the child. These conversation flows also served as scripts for our wizard-of-oz testing.

4.3.3 *Wizard-of-Oz Design Refinement & Testing.*

To refine the design of our voice-based interface, we conducted wizard-of-oz (WoZ) [47] testing using a 2nd generation Amazon Echo Plus, connected to an experimenter in another room using the drop-in feature on the Alexa smartphone app. This allowed the remote experimenter to speak through the device and hear the child's responses, much like a telephone call. We recruited 28 children ages 3–9 (M = 6.21, SD = 1.66; 17 female, 11 male; 8 in sibling pairs) at a local children's museum. The target age range was 5–8, but some sibling dyads included participants outside this range. Participants were brought into a private room and interacted with this wizarded system believing they were interacting with a functional game on the device. Through these sessions, we identified the importance of clear and useful fallbacks for invalid or misunderstood user responses, the need for providing explicit planning questions and response options in the Create a Story flow to support children unable to generate their own responses, and to give users time to think about their responses or confer with one another (leading to a press-to-speak, rather than open conversation, interface). We also used this testing to determine which flows to move forward with for implementation based on children's observed engagement levels and responses to follow-up questions.

4.3.4 *Implementation and Iterative Testing.*

Once we had stable and functional user flows from WoZ testing, we moved on to implementation of the StoryCoder app. We developed the app in Swift 5 for iOS 13 using Firebase storage and the Google Cloud speech-to-text and text-to-speech libraries. We chose to develop on iOS because it is the smartphone platform most readily compatible with these machine comprehension platforms. (Android was surprisingly incompatible with the Google Cloud libraries at the time of development.) Once we had functional prototypes for each flow, we continued with iterative testing to make refinements to the interactions and to test the components we were unable to test using the WoZ technique (e.g., visual selection components). Due to the COVID-19 pandemic, we conducted this iterative testing remotely over Zoom using screen-sharing and remote control of the iOS simulator. That is, children used the mouse or touchscreen on their device to interact with the simulator on the experimenter's device, and used speech interaction mediated by Zoom's audio. We tested the prototype with 22 children ages 5–8 (M = 6.78, SD = 1.01; 8 female, 14 male). This testing led to language refinement throughout the app, updates in any on-screen interactions, and general improvements to instructions.

4.3.5 *Evaluation.*

Upon completion of all user flows, we moved onto a short-term user study evaluation of the app's educational objectives and attitudinal impact, described in detail in the next section.

5 SHORT-TERM EVALUATION

To evaluate how well StoryCoder meets our objectives of introducing computational and storytelling concepts, we conducted a short-term evaluation focusing on children's performance on concrete in-app tasks, their ability to recall key concepts and recognize those concepts in new contexts, and their engagement with the app. Specifically we sought to answer the following research questions that relate to CT concept and storytelling knowledge, skills, and attitudes:

(1) Can children successfully apply the computing concept in practice and within context? (Computational Practice)
(2) Do children understand the computing concept being taught? Can they explain it? (Computational Recall)
(3) Can children recognize the computing concept in a new context? (Computational Recognition)
(4) Do children tell better stories with the scaffolding of this system? (Storytelling Practice)
(5) Can children recall the elements of the introduced story structure? (Storytelling Recall)
(6) Can children identify the story structure in new stories after using the system? (Storytelling Recognition)
(7) Do children perceive what they are doing as computational? (Attitudinal Perception)
(8) Do children find the game engaging? (Attitudinal Engagement)

These questions pertain to our overarching objectives of computational learning, development of story comprehension, and attitudinal shifts toward computing. They also serve to gauge the capacity of StoryCoder to foster growth in these areas in the long term, even after only a short term interaction. That is, a key aim of this study was to evaluate the validity of bridging CT with storytelling in creating a learning environment conducive to longer term educational outcomes.

Therefore, we measured children's ability to a) *use* introduced ideas in practice and in context (Computational and Storytelling Practice), b) *recall* those ideas (Computational and Storytelling Recall), and c) *recognize* those ideas in new contexts (Computational and Storytelling Recognition). This recognition is a "stretch goal" in a short term evaluation but is the foundation for learning transfer (i.e., a student's ability to apply taught concepts to a new problem), as children must first recognize the relevance of prior knowledge before they can apply it [5]. Finally, from a psychological standpoint, for an activity to positively shape a child's attitude toward computing in the long run, he or she must perceive it as related to computing (Attitudinal Perception) [34, 35], as well as engage positively with the system and the experience it creates (Attitudinal Engagement).

5.1 Participants

The study was conducted remotely over Zoom with 22 children ages 5–8 (M = 6.85, SD = 1.16; 11 female, 11 male; 7 entering kindergarten and 5 entering each of 1st, 2nd, and 3rd grade), who each participated in four 30- to 60-minute testing sessions on consecutive days. Children came from eight different U.S. states and had reading levels ranging from B (pre-reader) to M (early fluency). All children had no prior programming experience, were native English

Figure 2: A child participant interacts with the StoryCoder app installed onto an iPad during a Zoom testing session.

speakers, and had a phone or tablet device capable of running our software in their home. One additional child did not complete the study and is therefore not included in analysis. The Stanford IRB approved all procedures.

5.2 Metrics and Collected Data

We collected data to analyze practice, recall, and near transfer recognition of computing and storytelling concepts, as well as attitudinal metrics, described next. For each rubric-based metric, two coders first coded 20% of the data (four participants, chosen randomly by a random-number generator) and met to ascertain agreement before independently coding all of the data. We performed an inter-rater reliability analysis using the weighted Kappa statistic (quadratic weighting) to determine consistency among raters.

5.2.1 Computational Practice. To navigate through each user flow, the child must engage with a specific computational concept within the familiar context of storytelling. Therefore, to evaluate children's successful *application of CT concepts in practice and in context*, we captured their success rates in completing the assigned task, the number and types of mistakes made in doing so (i.e., speech-to-text machine comprehension errors or invalid input errors), and the amount of help they received from a parent or experimenter.

5.2.2 Computational Recall. In each flow, the user is first introduced to a new term (e.g., loops), receives an example of that CT concept, and then proceeds through the flow to further engage with that concept. To evaluate *recall*, we asked the child to describe what the new term means (e.g., *what do you think the word 'loops' might mean?*), explain how they used that concept in the game (e.g., *how did you use 'loops' in the game you just played?*), and give a novel example of that concept (e.g., *can you tell me an example of 'loops' from outside the game?*). After each question, the experimenter followed up with "Is there anything else you'd like to tell me?" one time before proceeding to the next question. Each question was scored on a scale from 0–2 according to a rubric (see supplementary materials), and scores for each concept question were combined for an aggregate concept recall score between 0–6. The inter-rater reliability for these scores was $\kappa = 0.88$ (95% CI, 0.88 to 0.88) for sequences, $\kappa = 0.84$ (95% CI, 0.70 to 0.98) for loops, $\kappa = 0.95$ (95% CI, 0.95 to 0.95) for events, and $\kappa = 0.93$ (95% CI, 0.89 to 0.98) for variables. We report mean and standard deviation of these scores.

5.2.3 Computational Recognition. Finally, for each computational concept introduced, we evaluated *near transfer recognition*. Learning transfer is notoriously difficult to achieve, and is the ultimate goal of education research. Therefore, despite the short duration of exposure in this study, we set transfer recognition (recognizing a taught concept in a new context—Scratch Jr. for our evaluation) as a stretch goal. Specifically, children were introduced in our task to "Johnny" (a fictional boy) who played a different app (Scratch Jr.) that taught the same four CT concepts in four games. Johnny would pick an animation to play with and change that animation a little bit through each game. Participants watched eight before and after animations (available in supplementary materials) to determine which concept/game Johnny was playing when he made the new animation. Four of these animations, which we considered near transfer, closely resembled the way StoryCoder presents those concepts (e.g., adding background music to an animation for the loops concept), whereas four were considered far transfer (e.g., repeating the entire animation multiple times for loops). We would expect a child who had learned nothing and had no prior exposure to the concepts to score at-chance (25%) on this recognition task. We therefore compare children's actual performance to this chance level performance to assess near transfer recognition.

5.2.4 Storytelling Practice. Similar to CT practice, we evaluate *storytelling practice* by examining participants' abilities to navigate through the story creation flow successfully. Additionally, the system itself provides two scaffolds for users: 1) specific constraints on the content of the story (e.g., the location) and 2) supportive prompts regarding the structure of the story. We evaluate the impact of this scaffolding by comparing children's stories told during gameplay to stories told in pre-testing during free generation (i.e., *"tell me a story about anything"*) and constrained generation of a story (e.g., *"tell me a story about a dog at a park that wants to get home"*). The stories were transcribed and then scored according to the Index of Narrative Complexity rubric [63], with an inter-rater reliability of $\kappa = 0.91$ (95% CI, 0.91 to 0.91).

5.2.5 Storytelling Recall. Also similar to CT recall, we evaluate the *storytelling concepts learned* during the Create a Story flow by asking the participant to recall all the important components (i.e., star, location, and problem) and parts (i.e., beginning, middle, and end) of a story as well as the relationships between them (e.g., the star is introduced in the beginning). We again transcribed and scored these responses, giving a point for each recalled concept, for a total score out of 10. Inter-rater reliability was $\kappa = 0.95$ (95% CI, 0.91 to 0.99).

5.2.6 Storytelling Recognition. Finally, to evaluate children's *recognition of story structure in new stories*, we leveraged the story structure evaluation task described by Stevens et al. [75]. In this task, a child listens to one of two stories before providing rubric-scored responses to a series of questions about the structural elements in that story. We pre- and post-assessed children on this activity, counterbalancing the order of the presented stories. It is important to note that story structure transfer was again a stretch goal, as the original study using these materials spanned a full academic year.

	Experimenter Help	Parent Help	Invalid Input	Machine Comprehension
Sequences	2.32 (2.59)	0.14 (0.64)	0.41 (0.96)	0.73 (1.24)
Loops	0.86 (1.28)	0.00 (0.00)	0.14 (0.38)	0.18 (0.50)
Events	2.05 (1.47)	0.00 (0.00)	2.33 (2.13)	1.00 (1.14)
Variables	1.05 (1.40)	0.23 (0.53)	1.18 (1.76)	2.50 (2.43)
Create	2.55 (2.86)	0.55 (1.14)	0.86 (1.17)	0.86 (1.08)
Listen	1.73 (1.35)	0.36 (0.95)	0.41 (0.80)	0.09 (0.29)

Table 1: Usability metrics for computing and storytelling practice, reported as mean (standard deviation). The first two columns denote the number of times an experimenter or parent assisted the child. Invalid Input errors, marked in the third column, denote the number of times children gave an input that was not understood by the system as a valid answer, whereas machine comprehension errors, in column 4, were the number of times a child gave a correct input but the speech-to-text was unable to understand what they said.

5.2.7 Attitudinal Perception. For children to show an attitudinal shift in favor of computing in the long run, they must first in the short-term recognize what they are doing as computational and see that activity as engaging [34, 35]. Given that the requirement that child participants be non-programmers precludes their ability to *directly* report if they view something as computational, we asked children to rank how helpful they thought practice with this game would be in each of the following school subjects: reading, math, science, technology/computer class, art, music, social studies, gym.

5.2.8 Attitudinal Engagement. Finally, to evaluate children's engagement levels with StoryCoder as a whole, we used the self-report based Giggle Gauge metric [21], which has been validated with children in our target age range and assesses engagement in terms of perceptions such as challenge, enjoyment, aesthetics, and endurability.

5.3 Procedure

Each participant completed four sessions over Zoom, where the first session was aimed at app setup and pre-assessments, the second and third sessions focused on app use, and the last session was set aside for post-assessments.

5.3.1 Session 1: App Setup and Pre-Assessments. In the initial session, we first completed the consenting process with the parent and child before installing the app onto the family's smartphone or tablet device of choice. In future sessions, the child played the game on this device while participating in a video chat so that we could see their reactions and monitor their progress. After completing this app setup, we used the remainder of the first session to build rapport with the child before conducting three short pre-assessment tasks (described in 5.2.4 and 5.2.6): free generation of a story, constrained generation of a story, and the story structure evaluation task from Stevens et al. [75].

5.3.2 Session 2: Create, Listen, and First CT Concept Flow. In the second session, the child began by creating a story and then moved to the listen flow, before playing one of the four computing concept flows (randomly chosen by the experimenter). After completing each flow, the child responded to the corresponding recall questions.

5.3.3 Session 3: Remaining CT Concept Flows. The Session 3 procedure mirrored that of Session 2, except that children played the remaining three computational thinking flows (again in a randomized order supplied by the experimenter).

5.3.4 Session 4: Post-Assessments. In Session 4, we began by quickly recapping the games played on the previous two days by showing children a short video of StoryCoder to remind them of what they did. We then post-assessed children's CT concept recognition in new contexts, evaluated their story structure recognition, measured engagement with the full app using the Giggle Gauge, and asked them to complete the attitudinal ordering task. (See 5.2 for a full description of these assessments.)

6 RESULTS

We report findings across all children. However, given the cognitive differences between the youngest and oldest participants in the recruited age range, we also report on the same analyses for the computing and storytelling learning metrics in an exploratory capacity across "younger" (5–6) and "older" (7–8) age groups.

All 22 children completed all user flows successfully, although two children faced text-to-speech problems during the events flow (one due to internet connectivity issues and one due to an early bug in the app), so we drop their data for that flow in these analyses. Children engaged with each flow for approximately 10 minutes.

6.1 Computational Metrics

6.1.1 Computational Practice. As shown in Table 1, children were able to navigate the loops flow rather easily, whereas there was a greater number of instances of experimenter help in the sequences and events flow. A main reason is that instructions in the sequences flow are relatively long and sometimes children would begin playing before they finished, thereby missing important parts. Once the experimenter prompted the child to carefully listen to the instructions again, the child was then typically able to proceed as with the other user flows. This issue could be solved with more concise instructions. In the events flow, on the other hand, children often gave invalid inputs by answering yes or no to an either-or question (*"Would you like to be given options for words or choose your own word?"*). The experimenter stepped in to clarify valid response options, but an update to this prompt and its corresponding fallback messages could resolve this issue in future iterations. Finally, while the variables flow had fewer experimenter interventions, it

	Younger	Older	Total
Sequences	2.58 (2.02)	4.50 (1.90)	3.45 (2.15)
Loops	3.25 (1.71)	4.70 (0.95)	3.91 (1.57)
Events	1.18 (1.08)	3.40 (2.50)	2.24 (2.17)
Variables	3.08 (1.88)	4.20 (1.14)	3.59 (1.65)
All Concepts	2.55 (1.85)	4.20 (1.74)	3.31 (1.97)

Table 2: Computing recall metrics, reported as mean (standard deviation), based on ability to describe the meaning of the term, usage of the concept in the game, and examples of the concept outside the game.

had more machine comprehension errors. Variables is the longest flow and the only one without multi-touch input options, meaning the child had to give voice input at all variable game states, unlike other computing flows that do allow for touch inputs at certain selection points (i.e., part for sequences, sound effect for events, and music for loops). The relatively high number of voice inputs in the variables flow thus likely contributed to more speech-to-text errors here. Considering the top N speech-to-text results, rather than just the top match could help reduce these problems. Overall, with a few exceptions where design solutions have been identified, children were able to navigate the app to engage with these computing concepts with little outside assistance.

6.1.2 Computational Recall. Furthermore, in just the short time children engaged with each flow, they were able to pull out at least a partial understanding of each concept, as shown in Table 2. Specifically, out of 6 possible points, children averaged 3.45 points on sequences, 3.91 points on loops, 2.24 points on events, and 3.59 points on variables, for an overall average of 3.31 out of 6 on computing concept recall. Note that recall for the events concept is lower than the other CT concepts, largely because multiple children who successfully navigated the flow defaulted to explaining social events (e.g., a birthday party) when asked for a definition of the word "events." Additional exposure or better clarification of the difference between these homonyms may improve the Events recall scores. Further breaking these scores down, we see that children score fairly consistently across the questions on definition ($M = 1.26$, $SD = 0.90$), usage ($M = 0.97$, $SD = 0.75$), and examples ($M = 1.08$, $SD = 0.96$), achieving, on average, partial credit on all of them. Overall, given that none of the children had any prior computing experience or exposure to these terms, this result is highly encouraging, suggesting that if children can grasp computing ideas after even a few sessions using a storytelling-based system, extended exposure could lead to a deeper, more comprehensive understanding of these CT concepts.

6.1.3 Computational Recognition. Finally, although learning transfer (i.e., usage of a concept in a new context) is a stretch goal for us, children were successfully able to recognize the new concepts in a novel context (Scratch Jr. animations, available in supplemental materials—see 5.2.3). On the eight recognition tasks, the proportion out of the total that children got correct was $M = 0.57$, $SD = 0.23$ (older: $M = 0.65$, $SD = 0.25$; younger: $M = 0.50$, $SD = 0.18$; see Figure 3). A single-tailed t-test on these scores confirms that children are performing significantly above chance on this transfer recognition task, $t(20) = 6.49, p < 0.001$, indicating their ability to recognize the CT concepts in a new task after just two days of playing with StoryCoder.

6.2 Storytelling Metrics

6.2.1 Storytelling Practice. All 22 children completed the create and listen user flows successfully, which is particularly notable considering that a subset of participants were unwilling to tell stories during the pre-assessments, when they did not have the app's guidance. There were more instances of experimenter involvement in the create flow than in other flows, although this is substantially skewed by two outliers in the older age group who received help 8 and 9 times, respectively, as they had longer discussions with the experimenter about their story plans (which could arguably be seen as more of a desired consultation than strictly needed aid).

Considering the quality of the stories created with StoryCoder, we saw that children told quantifiably better stories (based on the Index of Narrative Complexity rubric—see 5.2.4) when given experimenter-supplied constraints similar to those StoryCoder provides ($M = 13.82$, $SD = 6.92$) and when directly telling the story to the app during the scaffolded Create a Story flow ($M = 13.95$, $SD = 5.96$), than when free-telling a story ($M = 11.32$, $SD = 5.70$), though this difference is not significant when examined using a one-way, within subject ANOVA, $F(2, 42) = 2.723, p = 0.08$. However, when we separate by age we see that, while the younger children show only a negligible difference in story quality between conditions (free: $M = 10.67$, $SD = 6.10$; constrained: $M = 10.50$, $SD = 6.74$; in-app: $M = 10.42$, $SD = 4.08$), $F(2, 22) = 0.01, p = .99$, older children did tell significantly better stories when given story structure scaffolding (free: $M = 12.10$, $SD = 5.40$; constrained: $M = 17.80$, $SD = 4.89$; in-app: $M = 18.20$, $SD = 5.09$), $F(2, 22) = 9.71, p = .001$ (see Figure 4). A post-hoc pairwise t-test with a Bonferroni correction demonstrates that there is a significant difference for both constrained storytelling ($p = 0.03$) and in-app storytelling

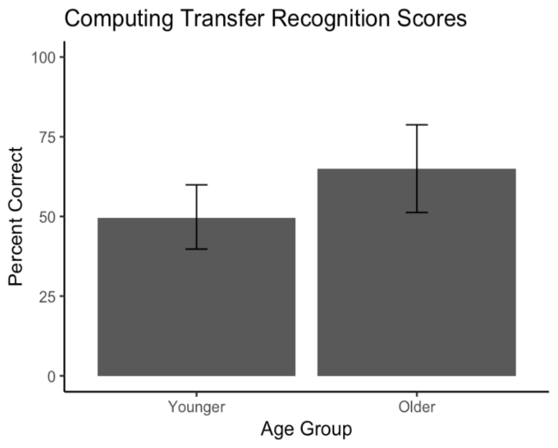


Figure 3: Younger and older children's mean post-assessment percentage scores on the computing transfer recognition task. Error bars show bootstrapped 95% confidence intervals.

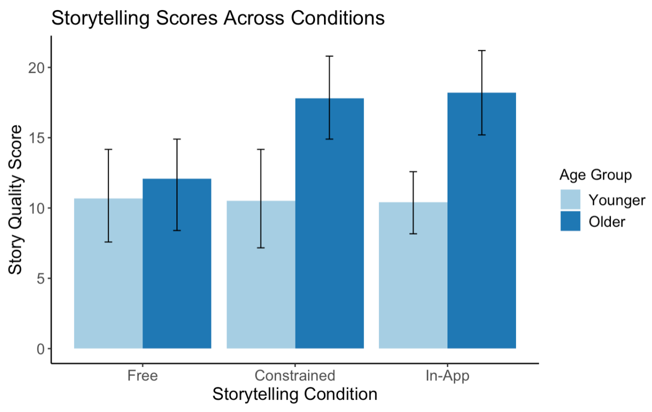


Figure 4: Younger and older children's mean scores on the free storytelling, constrained storytelling, and in-app storytelling activities. Error bars show bootstrapped 95% confidence intervals.

(p = 0.03), which elicit better stories from older children than free storytelling.

6.2.2 Storytelling Recall. Analyzing children's recall of story elements (M = 4.32, SD = 3.33), we also saw that older children (M = 6.40, SD = 3.20) similarly perform better than younger children (M = 2.58, SD = 2.35). Specifically, we observed that the younger children (many of whom were entering kindergarten and had no formal schooling at the time of the study) were largely able to identify the parts of a story (beginning, middle, and end), while the older children also recognized additional story components (e.g., character) or relationships between components and parts (e.g., that a *character* appears in the *beginning*).

6.2.3 Storytelling Transfer. Finally, for our story structure transfer task, we saw no difference in pre- (M = 10.54 SD = 5.23) and post-assessment (M = 10.41, SD = 4.52) responses in the free recall task, t(21) = 0.11, p = 0.91, for both younger (pre: M = 9.58, SD = 6.54; post: M = 8.42, SD = 4.12; $t(11) = 0.60, p = 0.56$) and older (pre: M = 11.70, SD = 2.98; post: M = 12.80, SD = 3.91; $t(9) = 0.87, p = 0.41$) age groups. This is not terribly surprising given this was a stretch goal for our short term study, as these evaluations are typically performed after longer (e.g., year-long) periods.

On the other hand, we were quite surprised by *decrements* observed for the prompted recall task (which immediately followed but was scored independently of the free recall, as per the procedure described in [75]), with analysis showing that children performed better in the pre-test (M = 9.45, SD = 3.17) than the post-test (M = 8.32, SD = 2.59), $t(21) = 2.41p = 0.03$. However, this finding did not hold when we separate the younger children (pre: 8.835, SD = 3.04; post: M = 7.42, SD = 2.23; $t(9) = 1.35, p = 0.21$) from the older children (pre: M = 10.20, SD = 3.33; post: M = 9.40, SD = 2.67; $t(11) = 1.96, p = 0.08$), likely given the decrease in power from a smaller N in each group. We believe this reduced performance is due to participant fatigue, as 1) the prompted recall task asked children to repeat information they had already provided during free recall and 2) children did so at the end of a 30 minute session during the pre-assessment, but at the end of a 60 minute session during the post-assessment. That is, children likely had more patience for such repetition during earlier portions of the study.

6.3 Attitudinal Metrics

6.3.1 Attitudinal Perception. To evaluate if children perceive StoryCoder as useful when it comes to learning about computing, we compared their rankings of the utility of the app for helping with a computer class, as compared to seven other school subjects. A Friedman test demonstrates that there is indeed a difference between the rankings of some of these groups ($\chi^2(7) = 40.60, p < .001$), with post-hoc analysis using a Conover test with Bonferroni correction showing that children rank StoryCoder as more helpful for computer class than art ($p < .001$), gym ($p < .001$), social studies ($p < .001$), math ($p < .001$), and music (p = 0.02). Computer class did not reliably rank higher than reading (p = 1.00), as expected, or science (p = 0.15), perhaps because most of the children involved in the study had not taken a formal science class yet so were uncertain about what such a class actually entails. Overall, we observed the following median rankings of school subjects (in terms of how helpful participants perceive StoryCoder will be in relation to other subjects): computing = 2, reading = 3, science = 4, music = 4, math = 5, art = 6, social studies = 6, gym = 7.

6.3.2 Attitudinal Engagement. Finally, we evaluated app engagement using the Giggle Gauge (M = 3.24, SD = 0.50), which found "moderate" engagement levels [21]. Specifically, the aesthetics subscore was relatively low (M = 2.95, SD = 1.07), which is as-expected given the voice-first design, but all other subscores (min: 3.17, max: 3.49) were in the moderate engagement range. This engagement score can likely be further improved through aforementioned design enhancements identified through our evaluation (e.g., shortening instructions and offering more useful fallbacks).

7 DISCUSSION

The results of our evaluation demonstrate that children—even those who were preliterate—were successfully able to navigate StoryCoder and engage with the presented computing concepts (Computational Practice), explain at least in part the meaning, the usage, and a novel example of those concepts (Computational Recall), and recognize those concepts in a near transfer situation (Computational Recognition). All children successfully told a story through StoryCoder, and older children were able to tell quantitatively better stories when they had the app's scaffolding (Storytelling Practice). Children were able to recall some elements of story structure (Storytelling Recall); however, we saw no evidence of children transferring new knowledge about story structures to novel stories (Storytelling Recognition). Finally, children perceived StoryCoder as being most helpful for a computer class (i.e., children do perceive the app as computational; Attitudinal Perception), and they demonstrated moderate levels of engagement (Attitudinal Engagement), suggesting the approach's longer term promise in supporting formal and informal connections with CS topics.

7.1 Supporting Multimodal Creative Engagement for Young Learners

Throughout the design and development of StoryCoder, we incorporated additional features, including and beyond those the paper has described, that were aimed at 1) exploring *voice* interfaces for educational content, 2) creating a more *personalized* learning experience for the child, and 3) supporting joint engagement for *co-creative* production.

7.1.1 Designing for Voice-Centered Interaction. First, we saw natural language dialogue as a desirable strategy to achieve our key accessibility, approachability, and engagement design goals. Voice interfaces allow presentation of complex or abstract content in a manner that does not require literacy, and a plain and simple interface reduces distractions and can help children stay on-task [16, 41]. Rather than relying on visual symbolic representations for these ideas, we saw how it is possible—and effective—to tell a child about concepts directly through speech and guide them through app usage with just-in-time dialogue. However, to make such spoken material accessible, we did need to carefully write content to use age-appropriate vocabulary, make viable response options evident, and ensure fallbacks were not repetitive. That is, we used very simple language (which we crafted and refined with children through our WoZ and prototype piloting sessions) for all dialogue, and we ended every output with a clear call to action in the form of an explicit question.

When children gave an invalid input or the machine comprehension failed, we provided a fallback response with randomized wording to reduce repetition that further clarified the possible response options and urged the child to try again. Children could receive such fallbacks up to two times for each decision point; on the third failed input attempt, the app randomly selects a valid response and gracefully proceeds through the flow. Moving forward, our 4-day evaluation surfaced additional improvements to make to instruction sets and decision points, for instance reducing the verbosity of instructions in the sequences flow and adjusting the wording (to clarify either/or questions) of prompts in the events flow in order to reduce confusion, increase the likelihood of eliciting operable responses, and ease interactions overall.

7.1.2 Supporting Personalization. The voice of the agent itself was another important design choice, given that children prefer personified voices [88] and humans extend gender-based stereotypes to agents based on their voices [58]. Most voice interfaces on the market (e.g., Siri and Alexa) have a default female-presenting voice, which has raised concerns about how these devices might reinforce gender biases and sexist behaviors [82]. On the other hand, we designed StoryCoder's agent to teach computing concepts, and a female role model can increase female student retention and performance in STEM fields [34, 53]. Therefore, rather than prescribing a gendered voice to the child, we included a voice gender option in the settings menu so that the child could pick the voice they preferred or that best aligned with their own identity. Similarly, we include an option to customize the color of StoryCoder's GUI. Given that the app is generally light on visual content, we added this feature to give children a sense of ownership and control over their interactions with the app, aligning with evidence from our formative investigation about children's desire for personalization.

7.1.3 Interactive Systems as Collaborators or Collaborative Mediators. Our formative work also illustrated children's desire to co-play with friends and family, and prior work has shown that children and families prefer games that they can engage with jointly [65]. Voice interfaces can pose a challenge in supporting such collaborative use, however, because systems often struggle to differentiate utterances aimed at the device from those targeted at a play partner. Therefore, to support joint engagement and allow play partners to discuss their plans and interactions with each other—and permit children time to think about their response—we created a microphone button (as can be seen in the bottom right corner of Figure 1's screenshots) to open the microphone stream, rather than keeping the conversation persistently open. In doing so, our system enables collaborators to communicate with one another without confusing the interface with unintended inputs. Indeed, across our WoZ and iterative design testing, we saw several sibling pairs conferring with each other, directing each other to supply input, and otherwise cooperatively engaging with the system to tell stories or decide on sound effects for those stories. Furthermore, to better support older siblings and parents as play-partners, we chose to include written transcripts of the system output on the screen, even though many of the target users were unable to read them.

7.2 Translating Short-Term Boosts into Longitudinal Benefits

Having completed our 4-day evaluation of StoryCoder and its voice-centered interactions, we would next like to explore how visual components might support children in gaining familiarity with and a more transferable mental connection to programming environments (e.g., conventional IDEs) they might encounter in the future [41]. In addition, there is merit in exploring the creation of visual analogies to reinforce the learning of abstract concepts without introducing distractions that take away from learning [16, 41]. For example, we can imagine representing different parts of stories as blocks that can fit together like puzzle pieces, reminiscent of those seen in block-based programming languages. Such metaphors also bear resemblance to conventional physical toys that children will already feel comfortable with and competent in manipulating, which may help to further boost self-efficacy in engaging with computational concepts, particularly for groups who face psychological barriers to access.

A key next step in our research agenda, after completing the proposed design updates, is deploying the app in a longitudinal study to examine learners' outcomes in-situ and over time. The proof-of-concept study presented in this paper provides strong evidence for the potential of StoryCoder and its interdisciplinary, integrative approach in stimulating learning and engagement even after a modest amount of exposure. However, an extended experiment that pre- and post-tests children's knowledge and CT skills is necessary to confirm sustained outcomes and deeply understand how such a system may fit into a child's everyday life, including as they grow and develop (e.g., advance in age, develop new hobbies and interests, and learn about new subjects in or after school).

A longitudinal evaluation would also better support the goal of measuring our approach's ability to shape children's storytelling skills, for which our short-term evaluation could understandably find only modest evidence. Importantly, participants in the current study used StoryCoder for just two days between pre- and post-assessments, with limited exposure to each flow. Given children in our sample had no prior programming experience, the exposure StoryCoder gave to this material was all novel; however, all children had prior experience with storytelling via books, fantasy play, and everyday social interaction. Key to our approach is the integration of interdisciplinary domains (computing and storytelling), but these differential levels of prior exposure and familiarity with the respective concepts is another important consideration for future work, including whether interface design choices should or could balance out these differences. A related point is the opportunity for systems to factor such baseline abilities into more personalized interactions based on a user's starting knowledge and custom goals they might have about particular learning topics.

7.3 Designing with Sensitivity to the Needs and Strengths of Young Users

Preliterate grade school age children are at a unique point: although they cannot engage with most traditional input mechanisms [22, 43], their verbal skills are relatively developed by kindergarten [3], their imaginations and capacity for creative activities are vast, and their voices are comprehensible to speech recognition software [87]. With block-based programming and programmable robots so dominant in the computational thinking education landscape, we sought to explore what such content learning might look like when separated from programming and instead embedded directly within a voice-based creative activity. We show early evidence of the efficacy of such a system and hope similar solutions can provide accessible, approachable, and engaging experiences for children to gain early exposure to computational thinking before moving to one of the more traditional programming-based learning tools later on in their education.

Taking together the natural abilities of young children, the technical feasibility of voice interfaces to handle their input, and our encouraging results that demonstrate how these capabilities can be effectively applied to the learning domain, we argue that voice-based, creative activities are a highly promising avenue for early educational technologies.

8 CONCLUSION

We presented StoryCoder, a voice-based smartphone application that leverages storytelling activities to introduce computational thinking concepts to children (ages 5–8) without requiring literacy skills. We discussed (1) our formative investigation and resulting design implications, (2) our wizard-of-oz studies and iterative design testing that guided creation of this system, and (3) the results of a multi-day, short-term evaluation study that demonstrated StoryCoder's efficacy in introducing targeted computational concepts while creatively engaging young users. Specifically, StoryCoder effectively introduced target computing concepts (i.e., sequences, loops, events, and variables), engaged children in an activity they perceived as helpful for later success in a formal computer class, supported older children in creating better stories than they did without the system, and provided younger and older children with the background to demonstrate above-chance performance on a near transfer post-assessment recognition task. Having identified improvements to be made to the system, including planned future work in multimodal design and longitudinal testing, our contributions establish the exciting potential for voice-centered creative learning activities to advance early computing education.

ACKNOWLEDGMENTS

We would like to thank Daniel L. Schwartz and R. Benjamin Shapiro for their advice and feedback and Thomas Hsieh for his programming contributions. Thank you also to the educators and families who participated in this research. This work was supported by the HPI-Stanford Design Thinking Research Program, The Brown Institute for Media Innovation, and a National Science Foundation Graduate Research Fellowship.

REFERENCES

[1] Marilyn Jager Adams. 1994. *Beginning to read: Thinking and learning about print.* MIT press.

[2] Isabel Machado Alexandre, David Jardim, and Pedro Faria Lopes. 2010. Maths4Kids: telling stories with Maths. In *Proceedings of the Intelligent Narrative Technologies III Workshop.* 1–6.

[3] American Speech-Language-Hearing Association. [n.d.]. How Does Your Child Hear and Talk? https://www.asha.org/public/speech/development/chart/

[4] Computer Science Teachers Association. 2017. Computer science standards. (2017).

[5] Susan M Barnett and Stephen J Ceci. 2002. When and where do we apply what we learn?: A taxonomy for far transfer. *Psychological bulletin* 128, 4 (2002), 612.

[6] Valerie Barr and Chris Stephenson. 2011. Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *Acm Inroads* 2, 1 (2011), 48–54.

[7] James F Baumann and Bette S Bergeron. 1993. Story map instruction using children's literature: Effects on first graders' comprehension of central narrative elements. *Journal of reading behavior* 25, 4 (1993), 407–437.

[8] Tim Bell, Jason Alexander, Isaac Freeman, and Mick Grimley. 2009. Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology* 13, 1 (2009), 20–29.

[9] Marina Umaschi Bers. 2018. *Coding as a playground: Programming and computational thinking in the early childhood classroom.* Routledge.

[10] Marina Umaschi Bers and Justine Cassell. 1998. Interactive storytelling systems for children: using technology to explore language and identity. *Journal of Interactive Learning Research* 9 (1998), 183–215.

[11] Aaron F Bobick, Stephen S Intille, James W Davis, Freedom Baird, Claudio S Pinhanez, Lee W Campbell, Yuri A Ivanov, Arjan Schütte, and Andrew Wilson. 1999. The KidsRoom: A perceptually-based interactive and immersive story environment. *Presence* 8, 4 (1999), 369–393.

[12] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada,* Vol. 1. 25.

[13] Jim Budd, Krystina Madej, Jenna Stephens-Wells, Janice de Jong, Ehren Katzur, and Laura Mulligan. 2007. PageCraft: learning in context a tangible interactive storytelling platform to support early narrative development for young children. In *Proceedings of the 6th international conference on Interaction design and children.* 97–100.

[14] Xiang Cao, Siân E Lindley, John Helmes, and Abigail Sellen. 2010. Telling the whole story: anticipation, inspiration and reputation in a field deployment of TellTable. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work.* 251–260.

[15] Justine Cassell and Kimiko Ryokai. 2001. Making space for voice: Technologies to support children's fantasy and storytelling. *Personal and ubiquitous computing* 5, 3 (2001), 169–190.

[16] Cynthia Chiong and Judy S DeLoache. 2013. Learning the ABCs: What kinds of picture books facilitate young children's learning? *Journal of Early Childhood Literacy* 13, 2 (2013), 225–241.

[17] Sue Inn Ch'ng, Yeh Ching Low, Yun Li Lee, Wai Chong Chia, and Lee Seng Yeong. 2019. Video Games: A Potential Vehicle for Teaching Computational Thinking.

In *Computational Thinking Education*. Springer, Singapore, 247–260.
[18] K-12 Computer Science Framework Steering Committee et al. 2016. *K-12 computer science framework*. ACM.
[19] Stephen Cooper, Wanda Dann, and Randy Pausch. 2000. Alice: a 3-D tool for introductory programming concepts. In *Journal of Computing Sciences in Colleges*, Vol. 15. Consortium for Computing Sciences in Colleges, 107–116.
[20] Griffin Dietz, James Landay, and Hyowon Gweon. 2019. Building blocks of computational thinking: Young children's developing capacities for problem decomposition. In *Proceedings of the 41st Annual Meeting of the Cognitive Science Society*.
[21] Griffin Dietz, Zachary Pease, Brenna McNally, and Elizabeth Foss. 2020. Giggle gauge: a self-report instrument for evaluating children's engagement with technology. In *Proceedings of the Interaction Design and Children Conference*. 614–623.
[22] Afke Donker and Pieter Reitsma. 2007. Aiming and clicking in young children's use of the computer mouse. *Computers in Human Behavior* 23, 6 (2007), 2863–2874.
[23] Stefania Druga, Randi Williams, Cynthia Breazeal, and Mitchel Resnick. 2017. " Hey Google is it OK if I eat you?" Initial Explorations in Child-Agent Interaction. In *Proceedings of the 2017 Conference on Interaction Design and Children*. 595–600.
[24] Allison Druin, Jamie Montemayor, Jim Hendler, Britt McAlister, Angela Boltman, Eric Fiterman, Aurelie Plaisant, Alex Kruskal, Hanne Olsen, Isabella Revett, et al. 1999. Designing PETS: A personal electronic teller of stories. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 326–329.
[25] Elisabeth Duursma, Marilyn Augustyn, and Barry Zuckerman. 2008. Reading aloud to children: the evidence. *Archives of disease in childhood* 93, 7 (2008), 554–557.
[26] Brittany Terese Fasy, Stacey A Hancock, Barbara Z Komlos, Brendan Kristiansen, Samuel Micka, and Allison S Theobold. 2020. Bring the Page to Life: Engaging Rural Students in Computer Science Using Alice. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. 110–116.
[27] Louise P Flannery, Brian Silverman, Elizabeth R Kazakoff, Marina Umaschi Bers, Paula Bontá, and Mitchel Resnick. 2013. Designing ScratchJr: support for early childhood learning through computer programming. In *Proceedings of the 12th International Conference on Interaction Design and Children*. ACM, 1–10.
[28] Willem Fontijn and Philip Mendels. 2005. StoryToy the interactive storytelling toy. In *Second International Workshop on Gaming Applications in Pervasive Computing Environments at Pervasive*.
[29] National Center for Education Statistics. [n.d.]. Children's Internet Access at Home. *The Condition of Education 2020* ([n. d.]).
[30] N Fraser et al. 2013. Blockly: A visual programming editor. *Published. Google, Place* (2013).
[31] David M Frohlich, Dorothy Rachovides, Kiriaki Riga, Ramnath Bhat, Maxine Frank, Eran Edirisinghe, Dhammike Wickramanayaka, Matt Jones, and Will Harwood. 2009. StoryBank: mobile digital storytelling in a development context. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1761–1770.
[32] Joanna K Garner and Cynthia R Bochna. 2004. Transfer of a listening comprehension strategy to independent reading in first-grade students. *Early Childhood Education Journal* 32, 2 (2004), 69–74.
[33] Jennifer W Glos and Justine Cassell. 1997. Rosebud: Technological toys for storytelling. In *CHI'97 Extended Abstracts on Human Factors in Computing Systems*. ACM, 359–360.
[34] Google. 2014. *Women who choose computer science–what really matters: The critical role of encouragement and exposure*. Technical Report.
[35] Google Inc. and Gallup Inc. 2016. Diversity gaps in computer science: Exploring the underrepresentation of girls, blacks and hispanics.
[36] Google Inc. and Gallup Inc. 2016. Trends in the state of computer science in U.S. K-12 schools.
[37] Google Inc. and Gallup Inc. 2017. Encouraging Students Toward Computer Science Learning.
[38] Lindsey Ann Gouws, Karen Bradshaw, and Peter Wentworth. 2013. Computational thinking in educational activities: an evaluation of the educational game light-bot. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*. 10–15.
[39] Mark Guzdial. 2015. Learner-centered design of computing education: Research on computing for everyone. *Synthesis Lectures on Human-Centered Informatics* 8, 6 (2015), 1–165.
[40] Yasamin Heshmat, Carman Neustaedter, Kyle McCaffrey, William Odom, Ron Wakkary, and Zikun Yang. 2020. FamilyStories: Asynchronous Audio Storytelling for Family Members Across Time Zones. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.
[41] Kathy Hirsh-Pasek, Jennifer M Zosh, Roberta Michnick Golinkoff, James H Gray, Michael B Robb, and Jordy Kaufman. 2015. Putting education in "educational" apps: Lessons from the science of learning. *Psychological Science in the Public Interest* 16, 1 (2015), 3–34.
[42] Michael S Horn and Robert JK Jacob. 2007. Tangible programming in the classroom with tern. In *CHI'07 extended abstracts on Human factors in computing systems*. ACM, 1965–1970.
[43] Juan Pablo Hourcade, Benjamin B Bederson, Allison Druin, and François Guimbretière. 2004. Differences in pointing task performance between preschool children and adults using mice. *ACM Transactions on Computer-Human Interaction (TOCHI)* 11, 4 (2004), 357–386.
[44] Felix Hu, Ariel Zekelman, Michael Horn, and Frances Judd. 2015. Strawbies: explorations in tangible programming. In *Proceedings of the 14th International Conference on Interaction Design and Children*. ACM, 410–413.
[45] Alan Kay. 2005. Squeak etoys, children & learning. *online article* 2006 (2005).
[46] Caitlin Kelleher, Randy Pausch, Randy Pausch, and Sara Kiesler. 2007. Storytelling alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1455–1464.
[47] John F Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)* 2, 1 (1984), 26–41.
[48] Varsha Koushik, Darren Guinness, and Shaun K Kane. 2019. StoryBlocks: A Tangible Programming Game To Create Accessible Audio Stories. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 492.
[49] Michelle H Land. 2013. Full STEAM ahead: The benefits of integrating the arts into STEM. *Procedia Computer Science* 20 (2013), 547–552.
[50] Tak Yeon Lee, Matthew Louis Mauriello, June Ahn, and Benjamin B Bederson. 2014. CTArcade: Computational thinking with games in school age children. *International Journal of Child-Computer Interaction* 2, 1 (2014), 26–33.
[51] Linda Liukas. 2015. *Hello Ruby: adventures in coding*. Vol. 1. Macmillan.
[52] Fei Lu, Feng Tian, Yingying Jiang, Xiang Cao, Wencan Luo, Guang Li, Xiaolong Zhang, Guozhong Dai, and Hongan Wang. 2011. ShadowStory: creative and collaborative digital storytelling inspired by cultural heritage. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1919–1928.
[53] David M Marx and Jasmin S Roman. 2002. Female role models: Protecting women's math test performance. *Personality and Social Psychology Bulletin* 28, 9 (2002), 1183–1193.
[54] Orni Meerbaum-Salant, Michal Armoni, and Mordechai Ben-Ari. 2011. Habits of programming in scratch. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*. ACM, 168–172.
[55] Robin Mello. 2001. Building Bridges: How Storytelling Influences Teacher/Student Relationships. (2001).
[56] Peggy J Miller. 1995. Personal storytelling in everyday life: Social and cultural perspectives. *Knowledge and memory: The real story: Advances in social cognition* 8 (1995), 177–184.
[57] James Moar. 2019. The Digital Assistants of Tomorrow.
[58] Clifford Nass, Youngme Moon, and Nancy Green. 1997. Are machines gender neutral? Gender-stereotypic responses to computers with voices. *Journal of applied social psychology* 27, 10 (1997), 864–876.
[59] Ageliki Nicolopoulou, Judith McDowell, and Carolyn Brockmeyer. 2006. Narrative play and emergent literacy: Storytelling and story-acting. *Play= learning: How play motivates and enhances children's cognitive and social-emotional growth* (2006), 124–155.
[60] Seymour Papert. 1980. Mindstorms: Computers, children, and powerful ideas. *NY: Basic Books* (1980), 255.
[61] Richard E Pattis. 1981. *Karel the robot: a gentle introduction to the art of programming*. John Wiley & Sons, Inc.
[62] Jackie Peck. 1989. Using storytelling to promote language and literacy development. *The Reading Teacher* 43, 2 (1989), 138.
[63] Douglas B Petersen, Sandra Laing Gillam, and Ronald B Gillam. 2008. Emerging procedures in narrative assessment: The index of narrative complexity. *Topics in language disorders* 28, 2 (2008), 115–130.
[64] Catherine Plaisant, Allison Druin, Corinna Lathan, Kapil Dakhane, Kris Edwards, Jack Maxwell Vice, and Jaime Montemayor. 2000. A storytelling robot for pediatric rehabilitation. In *Proceedings of the fourth international ACM conference on Assistive technologies*. 50–55.
[65] PlayScience and Casual Games Assosciation. 2016. *Let's coplay: A developer's guide to family cooperative mobile play*. Technical Report.
[66] Hayes Raffle, Glenda Revelle, Koichi Mori, Rafael Ballagas, Kyle Buza, Hiroshi Horii, Joseph Kaye, Kristin Cook, Natalie Freed, Janet Go, et al. 2011. Hello, is grandma there? let's read! StoryVisit: family video chat and connected e-books. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 1195–1204.
[67] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay S Silver, Brian Silverman, et al. 2009. Scratch: Programming for all. *Commun. Acm* 52, 11 (2009), 60–67.
[68] Elisa Rubegni and Monica Landoni. 2014. Fiabot! Design and evaluation of a mobile storytelling application for schools. In *Proceedings of the 2014 conference on Interaction design and children*. 165–174.
[69] Andy Russell. 2010. ToonTastic: a global storytelling network for kids, by kids. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*. 271–274.

[70] Kimiko Ryokai, Michael Jongseon Lee, and Jonathan Micah Breitbart. 2009. Children's storytelling and programming with robotic characters. In *Proceedings of the seventh ACM conference on Creativity and cognition*. ACM, 19–28.

[71] Daniel L Schwartz, Jessica M Tsang, and Kristen P Blair. 2016. *The ABCs of how we learn: 26 scientifically proven approaches, how they work, and when to use them.* WW Norton & Company.

[72] Dan Shapiro. [n.d.]. Robot Turtles.

[73] Arash Soleimani, Danielle Herro, and Keith Evan Green. 2019. CyberPLAYce—A tangible, interactive learning tool fostering children's computational thinking through storytelling. *International Journal of Child-Computer Interaction* 20 (2019), 9–23.

[74] Danae Stanton, Victor Bayon, Helen Neale, Ahmed Ghali, Steve Benford, Sue Cobb, Rob Ingram, Claire O'Malley, John Wilson, and Tony Pridmore. 2001. Classroom collaboration in the design of tangible interfaces for storytelling. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 482–489.

[75] Robert J Stevens, Peggy Van Meter, and Nicholas D Warcholak. 2010. The effects of explicitly teaching story structure to primary grade children. *Journal of Literacy Research* 42, 2 (2010), 159–198.

[76] Amanda Sullivan, Mollie Elkin, and Marina Umaschi Bers. 2015. KIBO robot demo: engaging young children in programming and engineering. In *Proceedings of the 14th international conference on interaction design and children*. ACM, 418–421.

[77] Lori Takeuchi, Reed Stevens, et al. 2011. The new coviewing: Designing for learning through joint media engagement. In *New York, NY: The Joan Ganz Cooney Center at Sesame Workshop*.

[78] Rivka Taub, Michal Armoni, and Mordechai Ben-Ari. 2012. CS unplugged and middle-school students' views, attitudes, and intentions regarding CS. *ACM Transactions on Computing Education (TOCE)* 12, 2 (2012), 8.

[79] Terrapin. [n.d.]. BeeBot. http://www.terrapinlogo.com

[80] Torben Wallbaum, Andrii Matviienko, Swamy Ananthanarayan, Thomas Olsson, Wilko Heuten, and Susanne CJ Boll. 2018. Supporting communication between grandparents and grandchildren through tangible storytelling systems. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[81] David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. 2016. Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology* 25, 1 (2016), 127–147.

[82] Mark West, Rebecca Kraut, and Han Ei Chew. 2019. I'd blush if I could: closing gender divides in digital skills through education. (2019).

[83] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.

[84] Jeannette M Wing. 2008. Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366, 1881 (2008), 3717–3725.

[85] Wonder Workshop. [n.d.]. Dash Robot. https://www.makewonder.com

[86] Ying Xu and Mark Warschauer. 2019. Young children's reading and learning with conversational agents. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–8.

[87] Ying Xu and Mark Warschauer. 2020. Exploring young children's engagement in joint reading with a conversational agent. In *Proceedings of the Interaction Design and Children Conference*. 216–228.

[88] Svetlana Yarosh, Stryker Thompson, Kathleen Watson, Alice Chase, Ashwin Senthilkumar, Ye Yuan, and AJ Bernheim Brush. 2018. Children asking questions: speech interface reformulations and personification preferences. In *Proceedings of the 17th ACM Conference on Interaction Design and Children*. 300–312.

[89] Eda Zhang, Gabriel Culbertson, Solace Shen, and Malte Jung. 2018. Utilizing narrative grounding to design storytelling games for creative foreign language production. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–11.