Designing ScratchJr: Support for Early Childhood Learning Through Computer Programming

Louise P. Flannery
Tufts University
105 College Ave.
Medford, MA 02155
louise.flannery@tufts.edu

Elizabeth R. Kazakoff
Tufts University
105 College Ave.
Medford, MA 02155
elizabeth.kazakoff@tufts.edu

Paula Bontá
Playful Invention Company
Montreal, Canada
bonta@media.mit.edu

Brian Silverman
Playful Invention Company
Montreal, Canada
bss@media.mit.edu

Marina Umaschi Bers
Tufts University
105 College Ave.
Medford, MA 02155
marina.bers@tufts.edu

Mitchel Resnick
MIT Media Lab
75 Amherst St.
Cambridge, MA 02139
mres@media.mit.edu

ABSTRACT

ScratchJr is a graphical programming language based on Scratch and redesigned for the unique developmental and learning needs of children in kindergarten to second grade. The creation of ScratchJr addresses the relative lack of powerful technologies for digital creation and computer programming in early childhood education. ScratchJr will provide software for children to create interactive, animated stories as well as curricula and online resources to support adoption by educators. This paper describes the goals and challenges of creating a developmentally appropriate programming tool for children ages 5-7 and presents the path from guiding principles and studies with young children to current ScratchJr designs and plans for future work.

Categories and Subject Descriptors

D.1.7 [**Programming Techniques**]: Visual Programming; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*User-centered design*

General Terms

Design, Human Factors, Languages, Theory

Keywords

Graphical Programming, Early Childhood, Education, STEM

1. INTRODUCTION

As digital creation tool increase in prevalence, early childhood education remains an area in which few educational technologies focus on digital creation or high-level thinking. Most computer programming tools target children at least 7-8 years old, but typically older. The plethora of mobile apps, software, and stand-alone technologies intended for younger children tend to focus on basic academic skills, such as recognition of letters and numbers, rather than content creation or high-level thinking. The ScratchJr project rests on the premise that children in kindergarten to second grade can indeed learn and apply concepts of programming and problem-solving to create interactive animations and stories. Needed to do so are powerful educational technologies and learning supports tailored for early childhood. Technical and curricular designs must be rooted in developmentally appropriate teaching practices sensitive to children's social, emotional, physical, and cognitive development [9].

Given the recent emphasis on the academic content of early childhood programs [18] and the importance of play in developmental and learning trajectories [9], digital technologies such as computer programming tools can bridge academic content with playful and meaningful activities [5]. Previous research has shown that children as young as four years old can understand basic computer programming concepts and can build and program simple robotics projects [3] [4] [6] [22]. Furthermore, early studies with the text-based Logo tools have shown that programming, when introduced in a structured way, can help young children with a variety of cognitive skills, including basic number sense, language skills, and visual memory [7]. The ScratchJr project described in this paper is designed to promote early childhood learning outcomes in well-established academic domains, such as literacy and mathematics, while also introducing children to computer programming and reinforcing problem-solving and foundational cognitive skills.

Three elements will comprise the ScratchJr project: 1) software designed with a developmentally appropriate interface and methods of interaction, 2) accompanying curricular materials as well as embedded opportunities to practice math and literacy content and other cognitive skills; and 3) an online community with resources for early childhood educators. This paper describes the educational goals and design principles of the ScratchJr project, prototype designs, and preliminary findings from research on children's use of ScratchJr. Foremost, this paper introduces the ScratchJr programming language and the story of its creation.

IDC '13, June 24-27, 2013 New York, NY, USA

2. BACKGROUND

2.1 Design Challenges

The early days of personal computing saw lively debate over the developmental appropriateness of technology in early elementary classrooms [8] [10]. Today, however, essential questions have shifted toward which digital tools to introduce and how to ensure that technology use supports and is supported by the broader curriculum [17]. Of course, computer programming may be difficult for novices of any age, and varied supports have been designed to facilitate this stage of learning programming [2] [14] [16] [19]. The developmental needs and capabilities of young children in particular pose additional design parameters [20].

Many salient issues influence the design of a programming environment for five to seven year old children. The strict syntax of text-based computer languages, such as Logo, can be unintuitive or frustrating for novice programmers. Graphical programming environments may simplify syntax difficulties but are often text-heavy, posing another challenge for pre- and early-readers. Hand-eye coordination and fine motor skills for controlling a mouse or touchpad to click and drag small elements around the screen can also impede effective use of software [13]. Beyond the technical aspects of reading and dexterity, young children's cognitive traits at different ages - or stages - follow patterns which vary significantly from those of older children and adults, beyond the scope of their knowledge [11] [12]. Self-regulation, systematic reasoning, working memory, and other, particularly high-level, thinking skills are still under development and change dramatically between ages five and seven [15].

2.2 Prior Solutions

Young children are intrigued by creating behaviors for animated characters but challenged by age inappropriate tools. A number of strategies have been attempted to resolve this disparity. Adults may sit with young children and give click-by-click instructions, although this scenario may pose both expected and counter-intuitive challenges for children's learning [1]. Simpler and more structured programming languages, such as ROBOLAB^{TM}'s PILOT level, or the Daisy the Dinosaur app aim to reduce difficult concepts for beginners. Other tools, such as the Bee-Bot or Cubelets robots or the ToonTastic animation app, preclude the need for programmatic scripts to create behaviors. These fascinating tools engage young children playfully with technology. However, their simplifications obscure important aspects of creative programming, such as sequencing a visible set of instructions, forming a program's flow-of-control which can then be evaluated and revised to produce a specific outcome.

Scratch, a free and widely used graphical programming language, was designed to make sophisticated programming accessible to children as young as eight and to people of all ages interested in a non-traditional entry to digital creation through programming. With Scratch, users program interactive art, stories, animations, simulations, and games by snapping together sequences of graphical blocks which represent instructions. Additional non-programmatic tools, such as a paint-editor and buttons for choosing and manipulating graphics and audio, round out the Scratch toolbox. In using Scratch, children learn core literacy and mathematical concepts in addition to valuable problem-solving skills [21].

As with most technologies designed to expand youth access to programming tools and knowledge, Scratch is not intended for children as young as five. The instruction set is large, relies on many complex concepts, and text labels, all barriers for young children to learn and use the interface. Many instructions rely on concepts children will not encounter or readily grasp until they are much older. Even for intuitive instructions such as "Move # Steps" or "Rotate #," unfamiliar units of measurement such as pixels or degrees are not easily or meaningfully learned by young children. Programmatically moving Scratch characters requires understanding of the Cartesian coordinate system, positive and negative numbers, and the relative size of numbers up to at least +/- 500. Expected only of older children and adults, the interface design's reliance on these concepts impedes young children's attempts to use Scratch. The implementation of some Scratch features and instructions also pose developmentally inappropriate challenges for young children, such as the lack of visible outcomes for many programming blocks or combinations of instructions. Other examples will be discussed in Section 4.1.2, on study findings.

Scratch has been a powerful and popular tool for introducing older children, teens, and adults to creative computation. Young children are equally interested in creating digital characters and stories and could benefit from doing so through programming. When presented with carefully designed tools and curricula, programming can be part of the growing set of digital creation tools for young children. Based on its history and popularity, Scratch is an excellent starting point for designing such a tool.

3. THE DESIGN OF SCRATCHJR

3.1 Developmentally Appropriate Design

The ScratchJr programming software currently in development by the authors is based on Scratch and intended for use by children in kindergarten through second grade (ages five to seven). This project aims to show how, given an environment designed for their developmental and learning needs, young children can reap many benefits from programming. The following principles guide the design of ScratchJr's features and affordances as a tool for young children to learn in many ways through programmatically creating interactive and animated stories. The implementation of each principle rests on translating developmentally appropriate expectations for the range of physical, cognitive, linguistic, and social-emotional traits among early elementary children into design features of the software to support learning and programming.

Low Floor and (Appropriately) High Ceiling Make it easy to get started with ScratchJr programming. Provide room to grow with concepts varying in complexity, but keep the tool manageable for the range of users.

Wide Walls Allow many pathways and styles of exploration, creation, and learning.

Tinkerability Make it easy to incrementally build up creations and knowledge by experimenting with new ideas and features.

Conviviality Make the interface feel friendly, joyful, inviting, and playful, with a positive spirit of exploration and learning.

Classroom Support Foster use in the classroom context and a wide range of learning outcomes though:

- Feasible management of use in classroom settings
- Support for building foundational knowledge which underlies multiple disciplines, such as sequencing, patterning, and iteration
- Support for discipline-specific knowledge from math, literacy, and classroom-selected curricula
- Support for problem-solving strategies and skills
- Complementary curricula and suggested teaching practices co-designed with early childhood teachers

3.2 Overview of ScratchJr

The ScratchJr software is comprised of a user's library of projects, a main project editor (Figure 1), and tools for selecting and drawing character and setting graphics. At the center of the project editor lies the story page, the scene under construction. New characters, text, and settings can be added by clicking large buttons labeled with icons: a cat silhouette, the letter A, and a mountain range, respectively. Up to four pages (thumbnails of which appear on the right-hand side) can be created and played in sequence as multi-scene stories. The set of characters included on each page is managed from the list to the left of the story page.

The blue palette of programming instructions lies along the center of the editor. Children display one instruction category at a time by clicking selectors on the left. Dragging instruction blocks from the palette into the scripting area below activates them. Snapping blocks together here creates programs that are read and played from left to right. The grid overlaid on the page provides a concrete, countable unit of measurement for characters' actions. It can be toggled on and off, as is most helpful to a given purpose, e.g. programming vs. presenting a project. The "Green Flag" ("Play") and red "Stop" buttons respectively start and interrupt the programmed animation. The page and the scripting area are seen as the most important parts of the activity and are visually emphasized through size, position, and color.

Both general and fine-grain details of the interface have been shaped by observations of children using Scratch and ScratchJr prototypes in small and large groups. The next section presents these studies and key observations made throughout. These findings, along with the principles for developmentally appropriate design noted in Section 3.1, guided the design decisions discussed in Section 5.

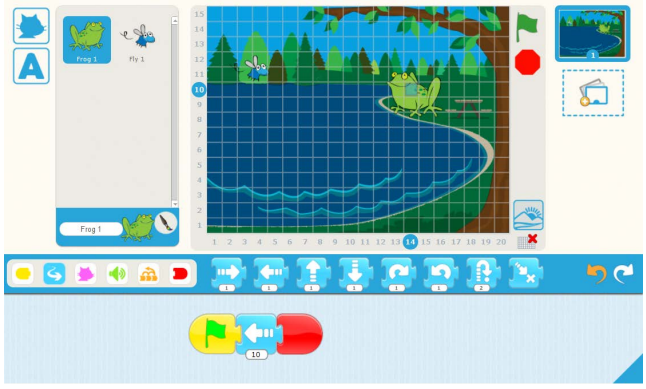

Figure 1: The main ScratchJr project editor.

4. RESEARCH STUDIES

Over the course of four research and development phases, the ScratchJr project aims to provide a graphical programming language for children in grades K-2, with curriculum materials and online resources for early childhood educators. The first phase provided baseline observations of young children using Scratch and teachers' thoughts on designing ScratchJr. An initial ScratchJr prototype was piloted along with supporting curriculum materials. During Phase 2, extensive data and observations of children's use of a revised ScratchJr tool are being collected in five classrooms. Unless otherwise stated, the observations noted refer to patterns seen among the vast majority of the children in each sample. These trends have informed further revisions to the software and curriculum. Detailed analysis of learning outcomes is ongoing; this section focuses on children's responses to the interface design at a higher level. Phase 3 will involve implementation in wider-spread classrooms and the introduction of an online resource community for teachers. After further revisions based on new findings, Phase 4 will see the public release of the software, curriculum, and online resources.

4.1 Phase 1: Baselines and Prototyping

4.1.1 Methods

Phase 1 research involved collecting baseline data on young children's use of Scratch (v1.4) and designing the initial ScratchJr prototype. In earlier work, Scratch use was studied with students (grades K-6) in a summer enrichment program. Groups of 15 children worked with 3 research assistants to complete an introductory curriculum and final project over five half-days. The nearly universal intrigue with creating animations and striking age-related differences in Scratch features learned led to follow-up studies.

More focused baseline data was collected from children and early childhood teachers. Approximately 40 children in grades K-2 worked in groups of 4 over four half-hour sessions to learn basic Scratch concepts and skills. About 30 of these children completed four additional half-hour sessions to apply and demonstrate their Scratch knowledge in a project. Nine pre-service teachers enrolled in a technology curriculum development course designed and led the projects. ScratchJr researchers documented observations of the children's use of Scratch and reactions to the curricula throughout.

Focus groups held with teachers rounded out the observations and recommendations collected prior to designing the ScratchJr interface and curriculum. After local teachers in the baseline data collection classrooms attended a workshop on Scratch and observed their students during the pilot studies just described, they made recommendations on ScratchJr features. In addition, non-local teachers who had used Scratch in their early elementary classrooms were contacted via the ScratchEd website and interviewed.

The final step of Phase 1, a ScratchJr prototype was implemented and evaluated in two pilot tests. First, 18 of the children in the baseline study children (ages 5-7), learned ScratchJr over three to four half-hour sessions. The second pilot, run as a summer enrichment program, involved 5 children and also tested a new ten-hour curriculum.

4.1.2 Findings

In conjunction with design principles, pilot work with the existing Scratch software allowed the identification of core elements and designs that ScratchJr should afford or avoid. For instance, in the pilot study, second graders had an easier time learning to find Scratch blocks based on their text labels than kindergartners or first graders, but literacy remained a barrier to developing proficiency with Scratch for most K-2 children. Systematically exploring the effect of numerical parameters on movement instructions was predicated on concepts not introduced until later elementary school: the Cartesian coordinate system, negative numbers, and relative size of numbers up to 500 or more. Skills ranging from understanding units of measurement to navigating hundreds of blocks to predicting the net result of a series of actions were beyond the grasp of kindergarten, first, and second grade students in the context of Scratch.

Some programming concepts - those with visible and intuitive outcomes - were accessible to these younger children. However, many Scratch instructions, either by themselves or when run in common groupings, do not have a visible outcome. This lack of feedback made it difficult for children to build associations between the instructions in their scripts and the actions that resulted. Furthermore, the fairly low-level instructions require the programmer to decompose an intended outcome into several levels of sub-components, whereas young children would succeed better if instructions and project components were fairly high-level and required fewer levels of decomposition of their goal.

While the open-ended possibilities of Scratch projects garnered children's attention at a high level, some functions' relatively unbounded designs proved distracting or confusing for young children, who explore and reason differently than older children. For instance, Scratch blocks allow number parameters dozens of digits long. Some children were drawn toward typing exceedingly large parameter values and away from the more rigorous task of determining a value to result in the originally desired action. This initially delightful foray tended to produce outcomes that children could not readily debug because of the size of the numbers involved and the nature of the tools to reset a project's state. Similarly, many children were drawn away from interest in creating a particular scene in favor of clicking blocks or tools that provided instant feedback but were not typically relevant to their goal. Young children who filled their screens with hundreds of characters with repeated clicks of the "mystery sprite" button neglected to then build behaviors for any of these characters. In spite of varied challenges, children were enthusiastic about the prospect of influencing the appearance and behavior of on-screen characters. The thrill of the instances in which they were able to do so demonstrated the niche that a program like Scratch could fill, if appropriately designed for use by children in grades K-2.

Tests of the initial ScratchJr prototypes found that many children made substantially more progress towards meaningful tinkering, learning of the software, and creation of mini-projects than they had with Scratch. For instance, the youngest (kindergartners) children, who primarily used the paint editor or filled their screens with random characters in Scratch, explored concrete ScratchJr programming functions (movement, appearance changes, sounds). Older (second grade) children learned enough to make sophisticated animated scenes in ScratchJr as year-end projects.

To ensure the software's appropriateness to the full range of children in K-2, some prototype features required further adaptation. It became clear that programming actions must not only have a visible output but also take time to run. This better facilitates exploratory learning of programming instructions' actions and supports children's perception of the role each action has in a program. During the summer ScratchJr sessions, a curriculum based on group and individual work with "ScratchJr Cards" to introduce core programming concepts and apply them in a semi-structured project was effective for that group of kindergarten and first grade children. A number of strategies were identified to support a wider range of children in managing the problem-solving process as they worked through each activity. From the interface color scheme to intuitive icon designs to the instruction set and its categorization, the two pilot tests provided rich feedback for further revisions to the software and curriculum in preparation for testing in full-sized and diverse classrooms.

4.2 Phase 2: Prototype Refinement

4.2.1 Methods

Phase 2 of the project, currently ongoing, focuses on the refinement of the ScratchJr prototype and the full introductory curriculum. The work takes place in five classrooms at two local schools: three kindergarten classes at one and a kindergarten and a combined 1st/2nd grade classroom at the other. The 100 children in the sample come from diverse cultural, language, and socioeconomic backgrounds; some receive special services in school. Four of the five classrooms have completed at least 9 sessions with a researcher-as-teacher, 1-2 research assistants, and up to 3 regular classroom teachers and assistants. Sessions last 30 to 60 minutes, depending on the needs of each classroom. Activities included introduction and demonstration of a new concept, group solving of a related programming challenge, and independent solving of the challenge, supported by a ScratchJr Card. In-class data was collected in the form of children's finished ScratchJr projects and video, audio, and screen-capture footage of each child at work.

Staggered introduction of the ScratchJr activities to the kindergarten classes at one school allows control-group comparison on pre- and post-curriculum assessments of foundational literacy knowledge. Baseline assessments of math and reasoning, collected at the other school, will inform future analysis of children's use of embedded math concepts and problem-solving strategies across grades. A revised curriculum will be tested next in a classroom where no ScratchJr activities have taken place. The other classrooms will also use the new curriculum format as they continue with activities on new programming concepts as well as tasks focused to highlight math and problem-solving skills. The classroom teachers, who are now familiar with ScratchJr, will become collaborators in redesigning, extending, and implementing the sequence of lessons and projects for their own classes.

Researchers are in the midst of documenting learning trajectories related to the interface and programming concepts as well as the use of foundational, discipline-specific, and problem-solving related knowledge and skills. The nature of the ScratchJr software requires some of these skills, while other skills are targeted by activities and projects; supports are designed in both the software and curriculum. Along

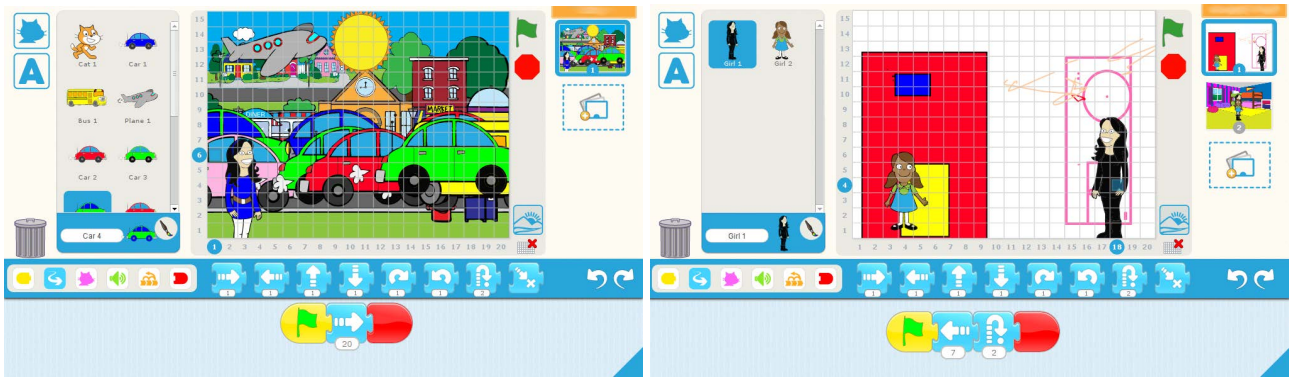

Figure 2: Kindergartners focused on different aspects of their projects: (Left) practicing new skills, in this case programming each of 11 characters identically, or (Right) creating their own graphics and refining programmed outcomes.

with further feedback from teachers and children, this ongoing analysis will inform further curriculum and prototype design revisions for use and re-evaluation during Phase 3.

4.2.2 Findings

These findings represent preliminary analysis of data collected during the first half of Phase 2. In four classrooms introduced to ScratchJr, children enthusiastically explored, programmed, and created animated scenes. Children were often on task with the curriculum, open explorations, and self-selected projects for 30-45 minutes, depending on interest and ability to work in a large group format. Off-task behaviors paralleled some seen with young children's Scratch work but proved less detrimental to goal completion. Children still spent extra time using the paint editor or adding characters they did not intend to program. In ScratchJr, though, these activities served to motivate rather than overtake the projects. Adults were primarily called upon to share exciting outcomes or help with non-obvious tasks such as deleting a character, spelling, technical issues, debugging, and higher-level management of approaching an activity. At this point, children have learned best how to:

- Find different categories of blocks
- Drag blocks to the scripting area and connect blocks to make a program
- Select characters and settings
- Edit or draw new characters and settings
- Add new pages
- Play their programs using the Green Flag Play button
- Use the Motion, Sound, and some Looks instructions to accomplish specific and general outcomes
- Use the End or Repeat Forever programming instructions to accomplish specific outcomes
- Explore unfamiliar blocks representing concrete and visible actions, to learn what they do
- Save their work and open new or existing projects.

Grade level differences in learning ScratchJr existed in the quantity of programming concepts and blocks learned over the same time period and in the complexity of concepts that were accessible. First and second graders tended to learn how to navigate the interface quicker than the kindergartners, allowing them to focus more on programming. Aspects of using ScratchJr which many children, particularly the kindergartners, have found challenging to learn in the context of the existing curricular activities include:

- Determining the specific function of some of the more sophisticated instructions, particularly meta-level instructions, which have no immediate visible outcome when run by themselves or in arbitrary combinations
- Understanding how to switch among several characters and program each one
- Coordinating multiple scripts within one character or across multiple characters
- Determining numeric parameter values to meet a goal
- Deleting unwanted characters (this was purposely made difficult at the time to avoid accidental deletions)
- Acquiring a common vocabulary of interface elements
- When stuck, choosing a problem-solving strategy that allows continued work on the task.

These observations are informing new revisions to the software and curricular approach. Children in any class represent a wide range of abilities that are useful in ScratchJr, from writing, icon recognition, and number sense to task management and flexible thinking. Providing differentiated or more scaffolded curricular activities will better ensure all children in grades K-2 an accessible entry point into ScratchJr programming.

4.3 ScratchJr in Action

This section presents examples of students' work from both self-directed and assigned activities to illustrate how children in the sample thought about and used ScratchJr after only a few sessions.

Figure 3: (Left) A kindergartner uses Motion blocks to recreate "Froggy Plays Basketball." (Right) A first grader uses speech bubbles and "Messages" to enact conversations from "Let's Say Hi to Friends Who Fly!"

Figure 2 shows examples of the range of project components that many kindergartners explored. In an introductory lesson on basic motions, the kindergartner who made the bustling city street scene (Figure 2; left) familiarized himself with new concepts as young children often do: by iterating on them over and over. In addition to programming an airplane to fly across the page, the assigned task, this child practiced adding characters and programming movement with eight individual cars, a school bus, and a girl on the sidewalk. He also discovered that moving characters a distance equal to the full grid width wraps them around the edges of the page and returns them to their starting points.

The pair of kindergartners who collaborated on a story about visiting a friend (Figure 2; right) devoted particular energy to drawing their own setting for the scene and producing specific outcomes through the characters' scripts. Their project includes a setting with each girl's house, drawn with rectangle, circle, and color tools. The two characters are programmed with the motions of walking down the street and greeting a guest. This required finding out how far to move the right-hand character to arrive precisely at the other house. The girls also added an instruction for this character to jump up and down in excitement over seeing her friend. The other character's script (unseen) mimics the motion of an elevator. Also visible on the right-hand side of the screenshot is a second story page, a scene of the visit itself. This project not only makes use of the programming concepts learned through structured lessons; it also relies on the children's ability to decompose a situation and map the story components to ScratchJr elements.

Following three to four introductory activities during which children explored and practiced core programming concepts and ScratchJr features, each classroom took on a unique project tying in other content under study, such as social studies or literature topics. The projects in several classrooms, including children in grades K-2, centered around children's authors being studied at the time. Students picked one or more story from which to retell a key scene or a beginning-middle-end sequence. These projects reinforce the composition and narrative flow of stories. Here are two examples comparing the complexity seen in the projects of children at different grade levels.

One kindergarten classroom made four-page ScratchJr stories depicting a day in the life of Jonathan London's "Froggy." In the story shown Figure 3 (left), Froggy "goes to space," "was little" and grows up, "goes swimming," and "plays basketball." The last of these scenarios is shown: the ball is programmed to travel up and into the child-drawn basketball hoop. The child who made this project pulls together multiple books to create his own narrative, although the sequencing does not follow a conventional chronological order of events. Projects also allow children to demonstrate the concepts they have learned comfortably enough to apply in a novel context. In this case, the characters are programmed with mostly motion instructions and the scripts make correct use the basic Start and End blocks that had been introduced.

Students in the study's mixed-age class (grades one and two) tended to create more complex projects given a similar task: retelling a chosen scene or story by the author Mo Willems. They programmed not only with motions, which were used in roughly half the stories, but also with instructions that are more challenging use or understand, from text to control flow. They also almost universally programmed and coordinated multiple characters. These features were much less common for kindergartners to implement without assistance.

Several of the most advanced students in this class used "Messages," a pair of instructions akin to procedure calls which make characters to carry out actions in turns rather than simultaneously. While about half the first and second graders used speech bubbles in their retellings, several combined speech bubble instructions with "Messages" to create alternating dialogue between characters. Several children also used multiple pages to illustrate the progress of the story, doing so in a more connected way than was typical of kindergarten stories. Figure 3 (right) shows a project with many of these observed trends. The cat on the first page (shown) says, "Hi" and the bird, whose program is unseen, will then reply and fly around the field, singing. The second and third pages present similar conversations with new characters, mirroring the flow of the original book. Equipped with more developed memories and reasoning abilities, the first and second graders had mastered basic navigation and programming concepts more quickly than kindergartners and had the time and cognitive resources to learn and apply more advanced concepts such as those described in this example.

4.4 Remaining Research

Many of the findings from the completed portion of Phase 2 studies have provided a sense of the timeline and supports that facilitate learning for children at different ages as they explore the range of ScratchJr tools. Upon completion of upcoming Phase 2 studies to test a curriculum with more built-in scaffolding and a variety of interface variations based on children's responses to the initial prototype designs, extensive analysis of the data collected throughout will add to the high-level findings presented here. Phase 3 will expand the pool of collaborating classrooms so data and feedback from educators and researchers in a wide range of classrooms can inform software prototype and curriculum revisions that best ensure that the designs are broadly successful. An online resource forum will also be piloted, evaluated, and revised. Following analysis of Phase 3 studies and final revisions to the software, curriculum, and online tools, these three components comprising ScratchJr will be released for public use, Phase 4.

5. DESIGN DISCUSSION

Having presented the goals of the project, the design principles underpinning initial ScratchJr prototypes, and key findings from observations of nearly 180 children using Scratch and ScratchJr, this section presents a more detailed discussion of the current design features of the ScratchJr software and curriculum and how they address the goals and challenges of creating ScratchJr as a developmentally appropriate programming environment for children ages 5 to 7. While building on many powerful affordances of the existing Scratch software, ScratchJr was extensively redesigned and differentiated to meet the needs and learning goals of kindergarten to second grade classrooms.

ScratchJr aims for children as young as five to meaningfully explore and gain independence in iteratively creating animated and interactive stories by considering the cognitive, physical, language, and socio-emotional traits of this age range during the design process. The layout is streamlined, with one third fewer programming blocks than in Scratch, only a few other essential tools, and no complex menus to navigate. Features are laid out intuitively and to minimize requisite mouse movement, clicking, and scrolling. Elements of the interface are large enough to facilitate targeting blocks and buttons with a mouse-pointer on a computer or with a fingertip on a tablet. Common strategies for building early literacy support children across the kindergarten to grade two range in navigating ScratchJr on their own. Labels are primarily icon-based so early readers can readily learn the elements of the tool. Any text is either child-created or can be revealed by the user, in which case the text is associated with the pertinent icon.

All features guide progress toward a project goal; they should be more useful than distracting. Across the software, the design of colors, layout, and functionality draw attention to programming functions. Manipulations of character graphics are implemented as programming blocks rather than non-programmatic buttons. Additionally, programming blocks must be placed in the scripting area to be functional, but once there, they can be clicked to run or attached to a trigger block, facilitating both tinkering and programming. Nearly all the instructions have visible outcomes and a long enough duration for children to observe; a select few control-flow or meta-level instructions are grouped together to minimize confusion for children not yet ready to explore them. This wide range of design choices aims to provide both physical and cognitive supports to help young children use ScratchJr with a level of independence that is reasonable to manage in a classroom setting.

5.1 Low Floor, Appropriate Ceiling

Tremendous variability in cognitive skills and knowledge exists among children in grades K-2. ScratchJr accommodates this range by providing an appropriately high ceiling, room to grow above a sufficiently unimposing entry point. There is variety in the complexity of instructions, from straightforward and concrete to more complex or abstract instructions, giving children appropriate content to explore as they master the basic tools. Teachers can therefore tailor a curriculum to the needs of the class. For example, the first structured activity in the current curriculum focuses on programming simple movement for a single character with start, motion, and end instructions. Later activities can include additional characters, more advanced uses of motion instructions, combinations of actions, and coordination of actions between characters.

While many aspects of ScratchJr can provide such a range of complexity, some tools and features afford less flexibility, and kindergartners and second graders may need incompatible designs. In these instances, a middle-ground solution was chosen as a starting point for testing, aiming to allow older children room to explore without shutting out younger children. For instance, children across early kindergarten to late second grade are comfortable with very different ranges of numbers. Although there are many possible solutions to this issue, the decision was made to set fairly low number parameter boundaries (under 25) in the initial ScratchJr prototypes. Second graders are ready to explore higher numbers, but the benefit to older children would be offset by the challenge posed to younger children.

5.2 A Tinkerable Space in Between

A tool with an efficaciously low floor and appropriately high ceiling must also provide a tinkerable space in between. Tinkerability is the extent to which a user can incrementally learn unfamiliar aspects of the tool and iteratively build up a project. If ScratchJr is sufficiently tinkerable, children in the target age range at different levels of ability or readiness should all find plenty of rich tools and functions to learn through exploration and creation. At the same time, they should not be hindered by designs and features implemented to address the needs of users at other levels. Many design decisions, particularly regarding what instructions to include, the behavior each block produces, and how to categorize the instruction set, were made to strike this balance.

The instruction set largely contains actions that children can readily grasp conceptually and whose actions are visible in the characters' behavior. These traits help children connect familiar knowledge to new and provide crucial feedback throughout exploration. As mentioned in Section 4.1.2, children learn new programming blocks more easily when they can see the connection between the block in the program and the ephemeral behavior of a character. To provide such feedback all action blocks take time to run and are highlighted in the scripting area as they do so. The limited number of instructions with instant or obscured outcomes minimizes

points of confusion, supporting tinkering and debugging. Other examples of designing for tinkerability by ensuring that embedded concepts are within grasp of the age range include the bounded range of numerical parameter values, concrete reference points such as a grid for counting units of distance, and allowing multiple ways of creating similar outcomes through combinations of blocks and/or manipulation of instructions' parameter values.

5.3 A Broad Welcome

ScratchJr was designed to be inviting and accessible to young children with a variety of styles of thinking and creating. The color scheme sets a playful tone while also drawing focus to programming tools and the created story page. The graphics are bright and whimsical, and the programmable actions are meaningful and fun. ScratchJr also supports multiple modes of engagement with a project. Children drawn to artistic endeavors can build investment in a project by designing characters and backgrounds. Children drawn to creating animations can explore the range of programming concepts systematically or through tinkering, based on personal styles of thinking, learning, and creating. The ability to create up to four independent "pages" and to integrate text and speech into a project allows children to create their own storybooks. For ease, children can use embedded graphics, or, to adapt and connect projects to their own lives and ideas, children can use simple scalable vector graphics editing tools to re-color existing graphics or design new ones. ScratchJr allows children's work to stem from a compelling mode of creation, whether artistic, programmatic, or narrative, and to proceed in ways that suit children's individual styles of creating and learning.

5.4 Supports for Classrooms

Designing ScratchJr for classroom use entails different constraints and considerations than informal after-school or at-home use cases, such as specific learning goals and a high ratio of young children to adults. ScratchJr has been designed to be feasible for both children and teachers to use in the classroom and to support three major categories of learning beyond programming itself: foundational knowledge structures, discipline-specific knowledge, and problem-solving.

5.4.1 Feasibility

The software was designed to be feasible for teachers to manage and usable for early elementary school children with minimal adult intervention. In contrast with other design principles, this goal was mainly met be ensuring that the net result of interface and interaction design decisions facilitated independent use by a large group of children. This was assessed in classroom studies by noting the frequency and purpose of children's reliance on adult support. These observations spurred several changes to the interface layout and interactions for using particular tools to maximize children's ability to discover new functions on their own. For example, an overly complex pathway for deleting a character was simplified after many children frequently asked for help to remember it. The use of ScratchJr Cards also was aimed at helping children independently work through each activity. Broadly, simple, intuitive designs and the overall tinkerability of ScratchJr allows for independent use and supports classroom feasibility.

5.4.2 Supports Foundational Knowledge

Foundational knowledge structures are those which apply across domains, skills such as sequencing, estimation, prediction, composition, and decomposition. Programming in ScratchJr can help children build these skills in several ways. Children explore the effects of sequencing as they compose each script from individual programming instructions matched to story or action components. As children use programming blocks' parameters, they develop skills in estimating, "How many?" or "How far?" Children are encouraged to predict what will happen when they run each refined iteration of their program, to think about whether the changes they have made will result in their intended outcome. The software then provides immediate feedback about the accuracy of their estimations and predictions. Making these situations explicit through curricular activities and encouraging frequent practice of them can help children develop these interdisciplinary and foundational skills.

5.4.3 Supports Discipline-Specific Learning

In addition to supporting domain-general skills, an important aspect of ScratchJr is to integrate opportunities for children to learn and practice concepts from state and national early literacy and math frameworks. ScratchJr facilitates learning discipline-specific knowledge in two ways. Core math and literacy concepts from national and state curriculum frameworks for grades K-2 are embedded and practiced through the basic use of the software. Teachers can also design ScratchJr activities which integrate and reinforce concepts from other domains under study by the classroom.

Several aspects of the programming project format reinforce literacy and book knowledge children are learning in their classrooms. Components of the software are discussed using familiar book and video analogies. The programming scripts are created and run from left to right, the same way as written English. To support narrative structure ScratchJr lets children create multi-page projects, like a book with a beginning, middle, and end. Text showing the name of each block can be revealed to support word recognition by letting children match intuitive icons with related text. Any other text that appears in a project has been created by the child. Though this paper has tended to highlight the building or "writing" mode of programming in ScratchJr, children also "read" and decompose scripts when they rebuild and try the example program on a ScratchJr Card, look over a peer's shoulder, or spend multiple sessions on a project and re-situate themselves each day.

Other aspects of the design focus on common math concepts for grades K-2, primarily number sense (magnitude) and measurement. Within these areas, children can pursue several pathways of quantitative exploration. Different kinds of measurement are included, e.g. distance, rotation, time, and iterations. Each one aims to have a visible and intuitive unit of measurement to facilitate exploration within this context. For instance, the removable grid of 20 by 15 squares mentioned earlier has been added to the page to give children a concrete unit to count and estimate as they determine how far to move a character. The designs of less straight-forward measurement units which do not have a structure similar to the grid to convey their meaning are still in progress, with simple solutions implemented to gather initial responses and assess children's understandings of the measurement concepts.

In using the grid, several possible strategies for programming movement of a given distance facilitates exploration of increasingly sophisticated number and programming concepts. To move a character three grid squares, a child could sequence three "Move 1" blocks, using the default parameter value for this block. The child could also use a single "Move 1" block and click the script or the Play button three times. Finally, the child could change the number parameter, creating a "Move 3" block. The grid's cells and numbered axes allow for strategies varying in complexity, from estimating and adjusting to counting to subtraction, for setting number parameters. The ScratchJr page is 20 units long, so children are encouraged to explore numerals whose magnitude is likely meaningful. In these ways, ScratchJr affords multiple pathways for exploring numbers and measurement in the context of a compelling task or project. Further work will investigate whether design variations, such as non-numeric labels on the grid axes, can better ensure that the tools provided for more sophisticated strategies like subtraction do not hinder or confuse children using simpler strategies like estimation.

Another priority of the project has been to make ScratchJr easily adaptable to individual teacher's and children's ideas, so several points for teachers to add their own content and project designs were incorporated. ScratchJr comes with a basic set of characters and setting graphics compared to the hundreds available in Scratch. This decision prevents the difficulty children have in navigating vast arrays of options, and also encourages children and teachers to create new graphics that might relate better to classroom-specific themes. Children can edit the included images or draw their own in an embedded scalable vector graphics editor. In one version tested, teachers can also import additional images and create lesson templates and project starters based on their curriculum goals at a given time. By providing flexibility to use the embedded assets and blank project layout or to add assets and provide a different starting point, teachers and children each have tools to make activities and projects relevant to classroom-based and personally meaningful content and learning goals.

5.4.4 *Supports Problem-Solving*

Also built into the software and the accompanying curriculum are supports for problem-solving: a) identification of a goal; b) formulation of a plan; c) development of an initial attempt to meet the goal; d) testing, evaluating, and sharing outcomes; e) debugging and revising each attempt based on feedback. Many of the design features described vis-à-vis a low floor and tinkerability also support problem-solving. They do so by reducing unnecessary low-level cognitive burdens, freeing up mental resources for high-level processes such as decomposing a story arc or troubleshooting a script that produces unexpected outcomes. Given fewer programming instructions than in Scratch, icon labels rather than text, and an emphasis on concrete instructions with visible, animated results, the challenge of learning to distinguish programming blocks and understand their functions is brought to the level of a five to seven year old without compromising the interesting and meaningful nature of possible programmed outcomes and projects. ScratchJr instructions were also selected to provide natural units of action to support decomposition of a story into programmed parts. A six-year-old child can more readily create a story about a frog in its habitat with instructions such as "hop" than if s/he needed to create each hop from lower-level instructions. These design decisions keep the challenge at an appropriate level and may help children devote sufficient cognitive resources to the many high-level thinking processes involved in imagining and creating a program. The ScratchJr Cards, a core element of the ScratchJr curriculum, also aim to support problem-solving by helping children map out the necessary components of a solution.

5.4.5 *Complementary Curriculum*

Created in collaboration with teachers, the ScratchJr curriculum targets a range of basic to advanced concepts. This curriculum provides teachers with a lesson format they can immediately implement and adapt as they are ready to create their own activities. The core introductory activities in the curriculum focus on different sets of concepts but follow a similar structure. First, the researcher or teacher leads warm-up activities including "Simon Says" variations, introduces and demonstrations the lesson topic, and leads the class in solving the challenge together. Then children complete the challenge using ScratchJr Cards (modeled after Scratch cards) as a guide. The activity provides a starting point for children to explore a particular programming concept or feature, and they may expand upon the lesson with their own ideas. The goals of the ScratchJr Cards and other curricular models being tested are to support children in tracking their own progress through the activity and to model ways of structuring complex problems. Some activities of the curriculum also encourage an open-ended approach to learning and creating. During their first exposure to ScratchJr, children freely explore the interface to see what it - and they - can do. After about five structured lessons, children apply their new skills over two to three sessions to a project that links to other classroom curriculum (e.g., an author study). By building on the software and curriculum designs, teachers have substantial influence on how to use and adapt ScratchJr as a tool.

6. NEXT STEPS

During Phase 1 and the first half of Phase 2 of the ScratchJr research project, the authors have collected baseline data, identified design principles, created a series of software prototypes, and conducted studies with nearly 180 children in informal and classroom contexts. Next, the classrooms will explore advanced activities and a new curriculum structure. Detailed analysis of the extensive data collected so far will shed further light on children's learning trajectories with ScratchJr programming and on the role of software and curriculum features in supporting learning.

During Phase 3, ten new classrooms will use ScratchJr. Goals of this phase include evaluation of the tools in a broader sample of classrooms and documentation of how teachers customize and implement ScratchJr curricula and manage the software's use in their classrooms. This phase will also explore tablet-based ScratchJr use, opening up new design criteria as well as insights into a more direct interface might impact children's learning. Final revisions to the prototype will be made following analysis of the data and feedback collected.

One area for future consideration relates to supports for teachers and classroom use of ScratchJr. Possible tools include those for monitoring student progress, managing graph-

ical and audio assets, and providing project starters to students. One support to be implemented during Phase 3 is an online ScratchJr forum with resources for educators and means to build connections among ScratchJr classrooms and teachers. Following broad classroom testing and revision of the software, curricula, and online resources during Phase 3, all three components of ScratchJr will be publicly released.

7. CONCLUSION

This paper has presented ScratchJr graphical programming environment, including the features and interactions designed for young children to create interactive and animated scenes and stories. Through the design of this software and accompanying curriculum materials, children involved in the project have engaged in age-appropriate computer programming and problem-solving while also building on traditional early childhood experiences: storytelling, numerical and spatial reasoning, creative thinking, and self-expression. The public deployment of the ScratchJr software, curriculum, and online community will provide young children with a powerful new educational tool as well as guidance for teachers and parents to implement it to the benefit of diverse areas of early learning, from math and literacy to interdisciplinary foundational knowledge structures.

Children in early elementary school classrooms will continue to create stories, art, and games with many different creative mediums. With a tool like ScratchJr at their fingertips, along with crayons and paper, they can engage in creative storytelling while also building creative problem-solving and digital literacy, skills seen as keys for participation and success in an increasingly digital world.

8. ACKNOWLEDGMENTS

This project is supported by National Science Foundation grant DRL-1118664.

9. REFERENCES

[1] L. Beals and M. Bers. Robotic technologies: When parents put their learning ahead of their child's. *J Int Learn Res*, 17(4):341–366, 2006.

[2] M. Ben-Ari. Constructivism in computer science education. In *ACM SIGCSE Bulletin*, volume 30, pages 257–261. ACM, 1998.

[3] M. U. Bers. Positive technological development: Working with computers, children, and the internet. *MassPsych*, 51(1):5–7, 2007.

[4] M. U. Bers. *Blocks to robots: Learning with technology in the early childhood classroom*. Teachers College, New York, NY, 2008.

[5] M. U. Bers. Using robotic manipulatives to develop technological fluency in early childhood. *Cont P on Sci Technol Early Child Educ*, pages 105–225, 2008.

[6] E. Cejka, C. Rogers, and M. Portsmore. Kindergarten robotics: Using robotics to motivate math, science, and engineering literacy in elementary school. *Int J Eng Educ*, 22(4):711–722, 2006.

[7] D. H. Clements. The future of educational computing research: The case of computer programming. *Inf Technol in Child Educ Ann*, 1999(1):147–179, 1999.

[8] D. H. Clements and J. Sarama. Strip mining for gold: Research and policy in educational technology – a response to "fool's gold". *Assoc Adv Comput Educ J*, 11(1):7–69, 2003.

[9] C. Copple and S. Bredekamp. *Developmentally appropriate practice in early childhood programs serving children from birth through age 8*. NAEYC, Washington, DC, 2009.

[10] C. Cordes and E. Miller. Fool's gold: A critical look at computers in childhood. 2000.

[11] D. H. Feldman. Piaget's stages: The unfinished symphony of cognitive development. *New Ideas Psychol*, 22(3):175–231, 2004.

[12] H. Gardner, M. L. Kornhaber, and W. K. Wake. *Intelligence: Multiple perspectives*. Harcourt Brace College, Fort Worth, TX, 1996.

[13] J. P. Hourcade, B. B. Bederson, A. Druin, and F. Guimbretière. Differences in pointing task performance between preschool children and adults using mice. *ACM T Comput-Hum Int*, 11(4):357–386, 2004.

[14] C. Kelleher and R. Pausch. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Comput Surv*, 37(2):83–137, 2005.

[15] C. Lightfoot, M. Cole, and S. R. Cole. *The development of children*. Worth, New York, NY, 2008.

[16] K. B. McKeithen, J. S. Reitman, H. H. Rueter, and S. C. Hirtle. Knowledge organization and skill differences in computer programmers. *Cognitive Psychol*, 13(3):307–325, 1981.

[17] NAEYC and Fred Rogers Center. Technology and interactive media as tools in early childhood programs serving children from birth through age 8. 2012.

[18] National Research Council. Mathematics learning in early childhood: Paths toward excellence and equity. 2009.

[19] D. A. Norman. *User centered system design: New perspectives on human-computer interaction*, chapter Cognitive engineering, pages 31–61. Lawrence Erlbaum, Hillsdale, NJ, 1986.

[20] C. Rader, C. Brand, and C. Lewis. Degrees of comprehension: Children's understanding of a visual programming environment. In *Proc ACM SIGCHI Human Factors in Computing Systems*, pages 351–358. ACM, 1997.

[21] M. Resnick. Sowing the seeds for a more creative society. *Learning and Leading with Technology*, 35(4):18–22, 2007.

[22] C. Rogers and M. Portsmore. Bringing engineering to elementary school. *J STEM Educ*, 5(3-4):17–28, 2004.