Using Upper-Elementary Student Performance to Understand Conceptual Sequencing in a Blocks-based Curriculum

Diana Franklin[§♮], Gabriela Skifstad[♮], Reiny Rolock[♮], Isha Mehrotra[♮], Valerie Ding[♮], Alexandria Hansen[♭], David Weintrop[§], Danielle Harlow[♭]

[§] UChicago STEM Education
University of Chicago
1100 E. 58th St
Chicago, IL 60637
dmfranklin@uchicago.edu
dweintrop@uchicago.edu

[♮]Department of Computer Science
University of Chicago
1100 E. 58th Street
Chicago, IL 60637
{gskifstad, rrolock, imehrotra, valerieding}@uchicago.edu

[♭] Gevirtz School of Education
UC Santa Barbara
Santa Barbara, CA 93106
akillian@umail.ucsb.edu
dharlow@education.ucsb.edu

ABSTRACT

As more elementary schools commit to integrating computer science instruction into their curricula, they seek guidance on what concepts are appropriate for students at different grade levels. Currently, little is known about how best to sequence computer science learning across elementary grades. In this paper, we present an analysis of 123 students' (age 9-12, grades 4-6) activities in a curriculum implemented in a visual block-based programming language. The goal of this work is to better understand the developmental appropriateness of foundational computer science ideas. All 4th, 5th, and 6th grade students in a single school completed the first module of a curriculum during the same school year with the same instructor. We analyzed each task students attempted and found that for simple concepts, there was little difference in performance between grade levels. However, differences were found for more complex topics, such as whether they completed initialization tasks and the way in which they solved 2-d navigation tasks. A closer look revealed that students understood the basic concepts, but were challenged by deeper applications of the basic concepts and influenced by non-computer science skills. This work serves as an empirically grounded investigation of elementary computer science learning and contributes to our understanding of computer science learning trajectories and concept sequencing in the late elementary grades.

1. INTRODUCTION

As more elementary school teachers begin to integrate computing into their curricula, they must design activities for students with disparate academic backgrounds and varying levels of prior computing experience. While standards are being released to articulate what concepts should be covered at what grade *band*, in order for teachers to effectively bring computing into their classrooms, further support is needed in two ways. First, goals need to be articulated by grade level rather than grade band. Second, we need to recognize that students often do not learn a concept completely in one year. Just as students learn to add numbers over three years in elementary school, students will revisit computing topics with more complex and in-depth exposure each year. Therefore, we need to understand how foundational computing concepts develop over several grades.

This paper presents a study of students across upper elementary grades (4-6) working through the same curriculum. By having students of different ages learn the same concept and work through the same set of activities, we can begin to understand grade-appropriateness of different concepts and the effectiveness of different types of instruction, as well as identify specific challenges they face. In doing so, we lay the foundation for a validated, grade-appropriate K-6 computer science curriculum that can start the next generation of learners on a path towards computing success.

In this work, we seek to answer the following two research questions.

- What computing concepts were challenging for students in different grades?
- Which non-computing concepts became stumbling blocks in projects intended to develop computing expertise?

We begin with a background on work investigating similar questions. We then describe our methods in Section 3. Section 4 presents our results, followed by a discussion in Section 5 and ending with a conclusion.

2. BACKGROUND

Computer science instruction in elementary school is an emerging field with many unanswered questions regarding designing age-appropriate curricula. Pertinent questions include what concepts should be covered at what ages? How deeply should each concept be covered? And, how do concepts align with or rely on non-computer science skills. In this paper, we attend to questions about content and relationship to non-computing skills.

Papert's work with the Logo language showed that programming was well within the cognitive abilities of elementary-aged learners [5]. Through working with late elementary aged students, Papert and colleagues found students were able to learn and use concepts such as sequence, loops, and conditionals. One of the more successful descendants of Logo is Scratch [8], a block-based, exploratory programming environment that gives students an intuitive interface as well as the ability to "remix" (copy and

SIGCSE '17, March 08-11, 2017, Seattle, WA, USA
 ISBN 978-1-4503-4698-6/17/03…$15.00
DOI: http://dx.doi.org/10.1145/3017680.3017760

modify) existing projects. The Scratch environment is widely used and has been found to be effective at engaging diverse and historically underrepresented learners in programming [7]. Despite widespread use, work towards understanding the cognitive affordances of the Scratch environment with this age group is only beginning to emerge. Seiter and Foreman [11] analyzed Scratch projects created by elementary-aged students to identify what blocks students used at which grades. Others' research of younger learners working with blocks-based tools discovered the need to teach learners about initialization [3] and to consider the user when authoring programs [4], two ideas taken for granted by educators working with older students.

Beyond computer science knowledge, programming projects are often dependent on non-computer science prerequisite skills such as mathematics knowledge, reading ability, and the ability to handle general cognitive load. Flannery et al. [2], in their description of designing Scratch Jr, identified that early elementary school students struggle with several mathematics concepts, as well as the overabundance of choices. Further, Hill et al. [6] found that the mathematics concepts were above grade level even for the advertised grades, and the students struggled as a result in both high-achieving and low-achieving schools. Seiter [10] similarly found evidence that overall academic performance profoundly affects success in computing. Fourth-grade students in a high-achieving school were able to complete projects with synchronization and actions in isolation, only faltering when the concepts were combined. Students in a low-achieving classroom, however, were unable to advance past the first project, showing a possible dependence between computing performance and performance in other subject areas.

3. METHODS

To answer our stated research questions, we took an iterative, design-based research approach to develop an age-appropriate blocks-based programming environment and accompanying curriculum. Design-based research is a systematic and flexible methodology that allows for collaboration between researchers and practitioners in real-world settings with the aim of improving educational practice [12]. This work involved a collaboration between educational researchers, computer scientists, and teachers who were using our curriculum and programming environment in schools. This collaboration often resulted in observations about what was and was not working in classrooms with students.

3.1 Materials

Students completed Module 1 of a curriculum using LaPlaya [6], a Scratch-like programming language and environment designed for 4th grade students. Module 1 is a project-based curriculum in which concepts taught are chosen to support the creation of a culminating digital storytelling project. Within each concept, there is a series of 3-4 tasks that students complete, each task slightly more complex than the last. It is intended that these tasks be completed in a single 45-minute work session. The lessons of Module 1, along with the number of tasks are shown in Table 1.

LaPlaya is a visual block-based language and environment inspired by Scratch but modified to be simpler for the younger end of the age range (Figure 1). Some blocks were modified to simplify the mathematics requirements (e.g. removing percentages, decimals, and negative numbers). In addition, the interface is configurable on a per-project basis, giving the curriculum designer control over aspects of the interface, such as which blocks, categories, sprites, scripts, and tabs are visible to the students for each task. Finally, each task has an automated analysis capability that a student can run to find out whether they have completed the task and, if not, get a hint as to what aspect is incomplete.

Table 1: Lessons and tasks in Module 1

Concept	Number of Tasks
Sequence, Interface	3
Breaking down actions	4
Event 1: On sprite clicked	3
Event 2: Other sprite clicked	4
Event 3: On key pressed	3
Initialization	3
X/Y Coordinates (optional)	5
Costume Changes	3
Scene Changes	3

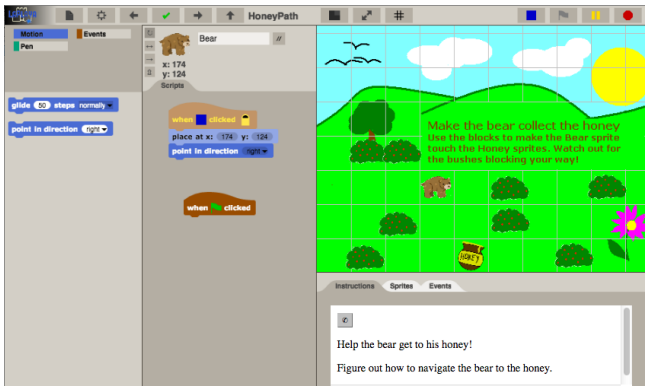

Figure 1: LaPlaya learning environment with a 2-dimension navigation puzzle directing the bear to the honey pot.

3.2 Data Collection

In the 2014-2015 school year, we tested our digital storytelling module and programming environment with over 1,500 students (ages 9-12) at 10 schools. For this paper, we analyze student work from one elementary school where all 4th, 5th, and 6th grade students completed the Module 1 curriculum (Table 1), totaling 123 students. The classes were required for students, and all sections were taught by the same technology teacher, thus controlling for teacher effects. Graduate student researchers (GSRs) attended most classroom meetings and recorded detailed field notes about student learning. In addition, GSRs collected video recordings of each class meeting, audio recordings of students asking questions, and interviews with teachers and students. Finally, digital artifacts including all tasks and final student projects were collected. After each class period, GSRs wrote analytical memos [9] that were shared with the research group. For this paper, we analyzed the digital artifacts collected.

3.3 Data Analysis

Our goal in data analysis was to identify what percentage of students demonstrated understanding and to identify particular challenges for students. For each task within an activity in Module 1, we wrote an analysis script to determine whether the artifact displayed understanding of the concept as well as identify the particular milestone of completion students reached. Each task's analysis code was tailored to that task.

To calibrate the analysis script, we first determined the minimum requirements necessary to "demonstrate understanding" of the concept. This is a lower bar than full completion of the task since these were learning tasks with repetition built into them, both of previously presented concepts and the current concept. Demonstrated understanding was determined as completing all scripts necessary for a single instance of applying that concept. Second, we identified specific struggles. We split progress into milestones and analyzed the code for reaching those milestones. Finally, a one-way analysis of variance (ANOVA) was conducted to compare the completion rates for each activity across grade levels, as well other variables related to specific tasks, to determine if the observed differences were statistically significant.

4. RESULTS

We present several findings from our analysis of the student tasks. We begin with the overall findings showing demonstrated understanding of several concepts. We then present more detailed results of several activities that show interesting behavior and/or differences between grade levels. Our results are broken down into three categories. We first present results that are gleaned from looking at how students did at a concept level. We then look more closely at how students completed tasks within a single concept. Finally, we analyze their final projects to see what concepts they used.

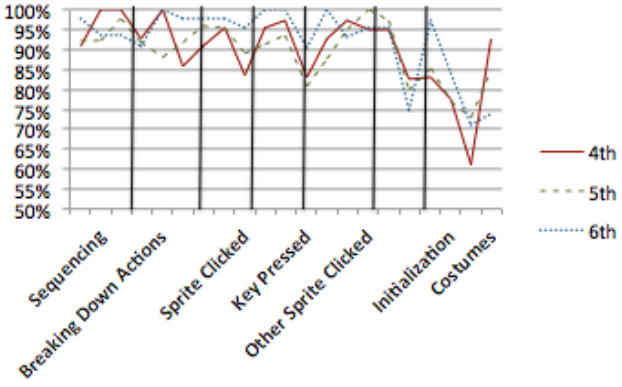

Figure 2: Percentage of students in each grade who demonstrated understanding for each task in the curriculum. Note: Y-axis begins at 50%.

4.1 Findings Between Concepts

We begin by presenting overall results for 4th, 5th, and 6th grade students. Figure 2 shows the percentage of students who demonstrated understanding (as defined by completion of one instance of the concept) for each task within each activity (note: the list of tasks can be found in Table 1).

While there were notable qualitative differences observed across grade levels for completion rates (see Figure 2), no differences at a statistically significant level were found. This is in part due to the small size of groups, and this analysis should be repeated on a larger sample to confirm or disconfirm the trends observed in Figure 2. Despite not resulting in statistical significance, the findings presented here are still useful for curriculum and interface developers working to engage children in learning computer science

Finding 1: Placing simple instructions in sequence and using simple events in a block-based language is accessible to 4th-6th grade students.

These results show that block-based programming environments, with projects using only a few blocks, are accessible to students in upper elementary school. Over 90% of 4th, 5th, and 6th grade students completed the first set of tasks, which involved ordering `glide to sprite`[1] blocks to draw simple pictures. Over 85% of all students completed the second set of tasks, which involved separately setting direction and moving a distance to navigate a simple 2-d grid.

In addition, tasks involving multiple events with just one action block per event were very accessible to all three grade levels. Fourth and fifth grade students struggled slightly on some tasks, but they attained over 80% completion on almost all tasks. The Other Sprite Clicked[2] activity had more tasks, so many students did not reach the last task, resulting in an anomalous dip. This shows that sequence and simple scripts using events are accessible to learners at the 4th grade level (and possibly even younger).

Finding 2: Initialization is challenging for 4th and 5th grade students.

The one concept in this curriculum that challenged students was Initialization. To initialize, students set the starting values for one or two attributes of a sprite. Differences between grade levels emerged, but, as stated earlier, the differences were not statistically significant. In general, 6th grade students still did well on these tasks, but 4th and 5th grade students struggled. There are several possible explanations including LaPlaya not providing enough scaffolding of computer science skills, not situating the challenge in a compelling context, or being too complex overall.

4.2 Findings Within Concepts

After artifacts were analyzed for completion rates, they were analyzed for the ways in which students solved the tasks. More nuanced differences between students of different grade levels emerged from this analysis.

Finding 3: 6th grade students are more precise at 2-dimension navigation than 4th and 5th grade students.

Breaking Down Actions comprised of tasks navigating a bear to a honey pot (Figure 1) while avoiding bushes in the path. The honey pot has a script that detects when it is touching the bear. Once the bear touches the honey pot, an animation of the bear with its nose in the honey pot occurs.

This project was simplified and scaffolded in response to results from the previous pilot year. As Figure 1 shows, a grid is drawn on the background. Each grid line is defined as 50 steps, and all sprites are aligned to the grid, simplifying distance calculation students must make to successfully navigate the terrain (all movements are multiples of 50).

One interesting attribute of this set of tasks revealed differences across the grades related to the precision of movement commands. Students could either program the bear to go to the honey pot precisely or overshoot it–the in-project detection would trigger an animation once the bear "touched" the honey pot. Therefore, a

[1]The `glide to sprite` block was added to LaPlaya specifically to create a very simple entry-level activity.

[2]The `on other sprite clicked` event was added to LaPlaya to remove the need for broadcast/receive messages to program an action in one sprite caused by a mouse click on another.

student could have programmed the bear to travel an inaccurately large distance and still receive positive feedback.

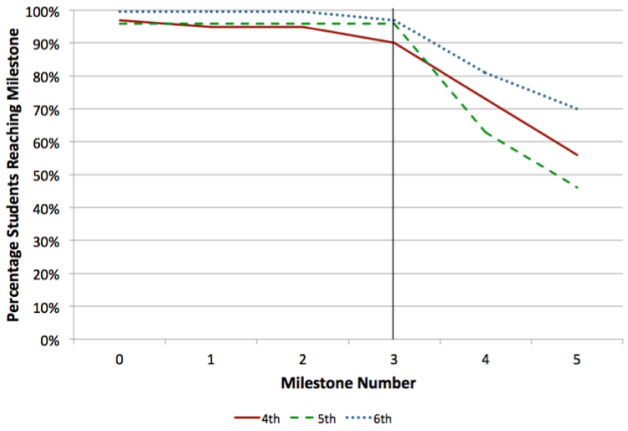

Figure 3: Percentage of students, by grade, who reached milestones for the last 2-dimensional navigation task.

Figure 3 shows the progress of 4th, 5th, and 6th grade students in completing the last task of the activity. Each point on the line represents the percentage of students who reached that level of completion. Milestone 3 corresponds to touching the honey pot, whereas milestone 4 corresponds to touching the honey pot with a relatively accurate measurement (stopping on the honey pot), and milestone 5 corresponds to touching the honey pot with an accurate and efficient solution (a single glide block calculated accurately). Most students reach milestone 3: 98%, 96%, and 90% of 6th, 5th, and 4th graders, respectively. Therefore, students easily solved the tasks to the level of reaching the honey pot, which satisfies the learning goals of this task.

Differences emerge, however, in inspecting the level of accuracy and efficiency of their solutions. Only 81%, 63%, and 73% of 6th, 5th, and 4th graders, respectively, measured the distance accurately (milestone 4). It seems counterintuitive that 4th grade students performed better than 5th grade students until we look more closely at their approach.

In order to better understand this trend, we performed a more detailed analysis on all students' solutions for tasks 2, 3, and 4 of Breaking Down Actions. We analyzed the artifacts for two factors. First, we categorized responses by the distance the bear was moved (accurate, approximate, or incorrect). Second, we analyzed whether the students used one or multiple `glide` blocks to complete a single leg of the trip. In our system, a single `glide` block's default distance was 50 steps, corresponding to one grid block. To move three grid blocks, one could place three `glide` blocks in succession, removing the need to perform mathematical calculations. The results from this analysis are presented in Table 2. In Table 2, the green and orange indicate contrasting performance for accurate calculations using one block. Yellow boxes indicate the two alternate approaches: accurate calculations using multiple blocks and approximate calculations using a single block.

Looking at the results for *accurate calculations* using *1 block* for each task, we can see that 6th graders were consistently more accurate and efficient than 4th and 5th graders (73% vs 56%, 86% vs 67% and 68%, and 70% vs 46% and 56%). These differences are highlighted in green (6th graders) and orange (4th and 5th graders). An interesting phenomenon occurs between 4th and 5th grade students. In Task 2, 30% of 4th grade students solved it by placing the correct number of consecutive glide blocks, whereas very few 5th grade students solved it this way. Instead, 5th graders appear to have attempted to calculate the accurate distance and this became became a barrier to completion. By Task 4, 4th grade students performed similarly to 5th grade students. In fact, comparing Task 4 to Task 2, more 5th grade students used consecutive blocks, and fewer 4th grade students did.

Table 2: Analysis of three tasks of Breaking Down Actions.

		Task 2			Task 3			Task 4		
		Acc	Appx	Inc	Acc	Appx	Inc	Acc	Appx	Inc
4th	1 block	56%	14%	0%	67%	2%	14%	56%	17%	7%
	n blocks	30%	0%	0%	12%	2%	0%	17%	0%	2%
5th	1 block	56%	20%	8%	68%	12%	8%	46%	29%	4%
	n blocks	4%	8%	4%	8%	0%	0%	17%	4%	0%
6th	1 block	73%	9%	0%	86%	5%	2%	70%	16%	2%
	n blocks	16%	2%	0%	5%	2%	0%	11%	0%	0%

Statistically significant differences were found only for Task 1. Fourth and fifth graders were significantly different for accurate completion using multiple blocks, $F(2, 142) = .262$, $p = .02$. In addition, 4th graders significantly differed from 5th graders for incorrect responses using 1 block, $F(2, 142) = -.08$, $p = .04$. Finally, 5th graders also significantly differed from 6th graders for incorrect completion using 1 block, $F(2, 142) = .08$, $p = .03$.

With this data we cannot say definitively what caused this discrepancy, but we speculate that this indicates that both 4th and 5th grade students struggled with the mathematical calculation for distance. They solved the problem in two ways–4th grade students were more likely to use the simpler method of using multiple `glide` blocks, whereas 5th grade students attempted the calculations and were incorrect more often.

These results indicate that 2-d navigation in increments of 50 are accessible to 4th, 5th, and 6th grade students, but the solution strategy may be different for 6th grade students vs. 4th and 5th grade students. For 4th and 5th grade students, it may be beneficial to first practice with 2-d navigation with grid blocks of 1 step before moving to 50-step grid blocks.

Finding 4: 4th grade students struggle to initialize multiple characteristics of a sprite.

In one particular Initialization task, students were asked to initialize two attributes of a cat: size and position. LaPlaya includes a separate initialization event (denoted with a blue square displayed alongside the traditional green flag) that the runtime environment enforces use of by disabling the green flag button until it is pressed. Students had already seen examples of initialization scripts as well as been asked to initialize position in previous tasks. This task introduced the need to initialize size. We can see in Figure 4 that most students initialized position properly (milestone 2). However, the number of students that properly initialized size (milestone 3) was lower, with 4th grade students showing the most significant drop. This means that more than 20% of 4th graders did not make any progress towards initializing a second characteristic of the sprite after correctly initializing a first feature, indicating they did not to apply their knowledge of initializing position and rotation to correctly define a starting state. This finding suggests that while initialization is appropriate for students as early as fourth grade, more complex initialization sequences that require coordination of multiple attributes or states might be more suitable for older students.

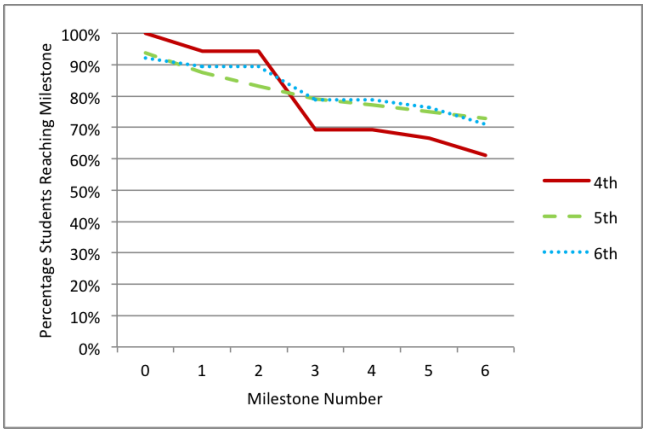

Figure 4: Percentage of students in each grade who reached milestones for the third Initialization task.

4.3 Culminating Projects

All students were given several programming sessions to complete a summative project. We analyzed 135 projects in terms of the total number of blocks, number of unique blocks used, median script length, and number of unique block categories used. The purpose of this analysis was to understand the scale of the projects (total blocks and script length) and diversity (unique blocks and unique categories) of computer science concepts incorporated by the learners across the three grades.

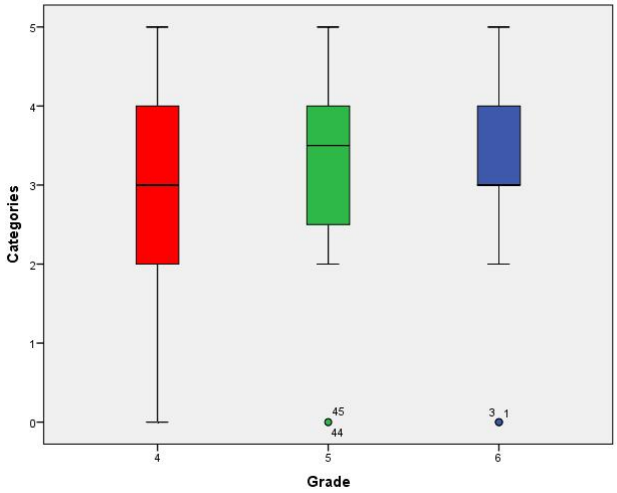

Figure 5: Number of block categories used in final projects.

Finding 5: There was little difference between grades in terms of total blocks used, median script length, and total unique blocks used.

In general, the projects were relatively simple. At least 20% of students in all three grades used no more than 9 blocks total and 4 unique blocks, while half of the students had median script lengths of less than 3. All three grades showed similar behavior, with only small differences, with the exception of diversity of categories, which we discuss below.

Finding 6: 6th grade students tended to use more categories (3), and the variance in the number of categories shrank as students aged.

Figure 5 shows the average number of categories used by students. This graph illustrates that while the average numbers are close, the variance decreases as students' grade increases. Further analysis revealed that 6th grade students used blocks from 3 unique categories more often than younger students: 84% of 6th graders versus 75% and 73% of 5th and 4th graders, respectively. This indicates that even if students, on average, perform similarly, perhaps the minimum standards for students at younger ages should be lower

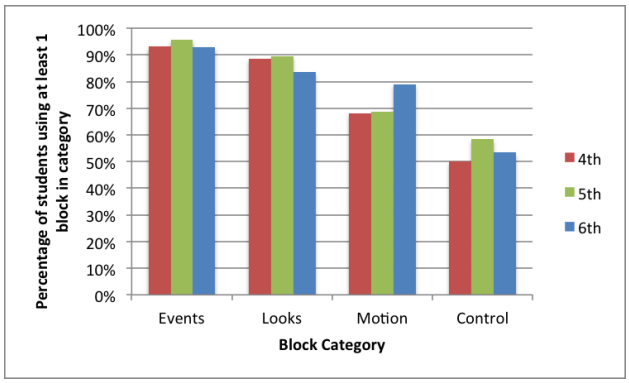

Figure 6: Percentage of students in each grade who used blocks from each category.

Finding 7: Students were most likely to use blocks from the "Events" category, followed by "Looks", "Motion" and then "Control."

We further analyzed final projects to find out *which* categories of blocks students used. Figure 6 shows the most commonly used category contained the Event blocks. This is not surprising given that an Event block is required in order for a script to be run. The most common Event block was the blue square block used for initialization. Thus, our requirement that an initialization event be pressed before the green flag, and the corresponding curriculum, likely succeeded in encouraging students to use this event. Other Event blocks were used fairly evenly. The second most common category was Looks, with `say` blocks (93 of the 123 final projects) dominating the block use, followed by `show/hide` (78/76) and `switch costume` (59). The third most common category was motion, where blocks that define movement in the X/Y coordinates were most common (83), followed by `glide in direction` (51) and `glide to coordinates` (30). Finally, among projects that used control blocks, the `wait` block was most popular (78), with only a few using loops (which were not taught explicitly in Module 1). In fact, more students used `wait` than `switch costume` (the only use taught in our curriculum) indicating that students transferred their knowledge of `wait` to other blocks.

Overall, this suggests that specific blocks in the Looks category are simpler for students than Motion blocks. Only for 6th grade students was the use of Motion blocks similar to that of Looks. This could be related to familiarity with the coordinate plane and better performance on the 2-d navigation puzzles, though it is unknown whether or not the relationship is causal. In addition, the idea of initialization is accessible, but, as we saw in Section 4.2.2, the details are challenging to get correct.

5. DISCUSSION

This work provides several insights that can be applied to activity development in upper elementary school grades. First, sequence and simple events are very accessible to young students. This finding reinforces design decisions made by other environments designed for younger learners, such as Logo and Scratch Jr. and

adds empirical evidence towards the design of curricula for elementary students focusing on movement-based activities (or "turtle graphics" activities in Logo parlance).

Second, activities involving precise mathematical calculations can result in undesirable difficulty and potentially be a barrier for some students toward engaging with the underlying computer science concepts. In this work, we found that while all students were able to give navigational commands to their sprite, when students needed to use precise mathematical calculations, younger learners performed worse than older learners. Our analysis showed the difference was attributed to mathematical aspects of the task rather than difficulty with the concept. The important take away from this finding for curriculum designers is to try to minimize the external mathematics required for activities as it may introduce an unintended barrier to the desirable computer science content of the lesson.

A third discussion point from this study is related to our investigation of initialization. This work reveals a developmental difference in terms of initializing a single attribute of an object and initializing multiple attributes. One might assume that initialization is an atomic idea that a student does or does not understand. However, this work shows there to be more to the concept of initialization of sprites for younger learners. The implications of this are twofold. First, when teaching initialization or designing initialization activities, it is important to design in a trajectory from simple to more complex tasks so students better understand the concept of initialization. Second, and more importantly, this shows the need to carefully analyze all aspects of introductory computer science instruction and not take for granted any part of the act of programming. It is easy for an expert programmer, or even just a casual programmer, to think of initialization as an atomic unit that students either do or do not understand. We suspect that there are other aspects of programming that share this characteristic and plan on further investigating them as future work.

Finally, this work is further evidence of the existence of developmentally appropriate computer science instruction and shows that there are concepts more and less suitable for students across grades 4, 5, and 6. In conducting this work, we can begin to tease apart differences in these early stages of computational learning and use them to inform age-appropriate instruction. At the same time, these findings provide clues for the development of a larger K-6 computer science trajectory, another avenue of future investigation we intend on pursue to more fully map out what effective computer science instruction might look like in upper elementary school.

6. CONCLUSION

This paper presented detailed analysis on how students progressed through an upper-elementary computing curriculum. We identified several insights, applied both within and across concepts that can guide development of effective K-6 computer science instruction. It is through studies such as these that we can gain knowledge necessary to create curricula aimed at a broad set of students' developmental and academic levels, as well as the ability to provide differentiation and accommodation for individual students.

7. ACKNOWLEDGMENTS

This work is supported by the National Science Foundation CE21 Award CNS-1240985. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not necessarily reflect those of the National Science Foundation.

8. REFERENCES

[1] Clements, D.H. and Battista, M.T. 1989. Learning of geometric concepts in a Logo environment. *Journal for Research in Mathematics Education*. (1989), 450–467.

[2] Flannery, L.P. et al. 2013. Designing ScratchJr: Support for early childhood learning through computer programming. *Proceedings of the 12th International Conference on Interaction Design and Children* (2013), 1–10.

[3] Franklin, D. et al. 2016. Initialization in Scratch: Seeking Knowledge Transfer. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (2016), 217–222.

[4] Hansen, A. et al. 2016. User-Centered Design in Block-Based Programming: Developmental & Pedagogical Considerations for Children. (2016), 147–156.

[5] Harel, I. and Papert, S. 1990. Software design as a learning environment. *Interactive Learning Environments*. 1, 1 (1990), 1–32.

[6] Hill, C. et al. 2015. Floors and Flexibility: Designing a programming environment for 4th-6th grade classrooms. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (2015), 546–551.

[7] Maloney, J.H. et al. 2008. Programming by choice: Urban youth learning programming with Scratch. *ACM SIGCSE Bulletin*. 40, 1 (2008), 367–371.

[8] Resnick, M. et al. 2009. Scratch: Programming for all. *Communications of the ACM*. 52, 11 (Nov. 2009), 60.

[9] Saldaña, J. 2015. *The coding manual for qualitative researchers*. Sage.

[10] Seiter, L. 2015. Using SOLO to Classify the Programming Responses of Primary Grade Students. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (New York, NY, USA, 2015), 540–545.

[11] Seiter, L. and Foreman, B. 2013. Modeling the Learning Progressions of Computational Thinking of Primary Grade Students. *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (New York, NY, USA, 2013), 59–66.

[12] Wang, F. and Hannafin, M.J. 2005. Design-based research and technology-enhanced learning environments. *Educational technology research and development*. 53, 4 (2005), 5–23.