Identifying Computational Thinking in Storytelling Literacy Activities with Scratch Jr.

Prof. Tony Andrew Lowe, Purdue University, West Lafayette

Tony Lowe is a PhD student in Engineering Education at Purdue University. He has a BSEE from Rose-Hulman Institute of Technology and a MSIT from Capella. He currently teaches as an adjunct at CTU Online and has been an on-and-off corporate educator and full time software engineer for twenty years.

Dr. Sean P. Brophy, Purdue University, West Lafayette

Dr. Sean Brophy is the director of Student Learning for the INSPIRE Pre-college Research Institute at Purdue University. His research in engineering education and learning sciences involves developing young children's cognitive ability to think and reason during complex problem solving activities. As part of this research he explores new methods to enhance informal and formal learning experiences using technology.

Identifying Computational Thinking in Storytelling Literacy Activities with Scratch Jr. (RTP)

Abstract

Students are learning to interact with, design for, and sometimes even program computers at earlier and earlier ages. Teachers and researchers can relatively easily measure progress in learning programming tasks, but assessing conceptual understanding of computation, mainly when programming tasks are not present, is less defined or non-existent. Computational Thinking (CT) generally refers to knowledge and skills apart from, and possibly a precursor to, the ability to write computer programs, yet is commonly measured through the quality of programming. Are there ways of qualifying CT 'maturity' outside of programming tasks?

This study looks at the intersection of CT and CS in first-grade learners who are developing computational solutions involving literacy tasks. Students retell a story by animating characters in Scratch Jr. by breaking down the story, creating an animation storyboard, and finally implementing the plan in Scratch Jr. For most of the participants, this is their first time using Scratch Jr. or any programming language. Therefore, their early experience with technology means they are working on an analysis of a story using literacy skills, considering a visual representation of the story, and learning how to realize the expression of the storyline using a computer language. To better understand the general concept of CT, we analyzed video data of the students building their animated story, along with the artifacts of design and code. This qualitative data provides indicators of how their thinking progresses from little to at least working knowledge of writing a program. Using qualitative analysis, we evaluate how their initial design evolves, simplifies, or otherwise changes as the novices move from general problem solving to creating a computational solution. CT seems to be a specialized version of design and engineering design in general. Understanding how these first graders move from generally approaching a problem to a computational solution may give insight into how CT matures and the role of programming in that maturity. By capturing how a design evolves, we can suggest how CT can be 'seen' in artifacts produced that are not programming code. The goal is to find markers of CT in design and language that transcends specific programmed implementations.

Introduction

As the research into Computational Thinking (CT) snowballs, there is a split into the role of programming in teaching CT. Some frameworks dive into programming concepts (Brennan & Resnick, 2012; Grover & Pea, 2013; Lowe & Brophy, 2017) while others are content with describing general ideas of problem-solving with (or some without) computing devices ("Computational thinking concepts guide," n.d.; Selby, 2013; Shute, Sun, & Asbell-Clarke, 2017). Researchers and educators generally agree that the goal of CT is to allow a wider audience than just professional programmers to engage with and be part of creating the computational devices our lives have come to depend upon. For young children, this means both developing precursor CT skills as well as an identity that strengthens their belief they can participate in creating technical solutions. The way we define CT is essential as it drives funding, research, and pedagogy. The definition of CT should promote both positive attitudes towards computing as well as genuinely building precursor skills for a literate workforce capable of using computational media. How does the introduction of technology change problem-solving strategies, and what is the mix required to enable CT, yet not expect everyone to be a programmer? This paper observes how the stories children tell change in light of learning how ScratchJr enables animation and considers the role of programming in learning CT.

Literacy involves the competencies associated with reading, writing, listening and speaking, and interpretation of literature (Common Core State Standards Initiative, 2010). Literacy is critical to learners developing the ability to understand literature, identify significant events and articulate those events. The medium in which children hear a story can impact their ability to retell stories. For example, adults reading the story with children viewing the static pictures is a radically different presentation than animated video. Video provides learners with dynamic visual support that helps them build mental models of the characters and their actions (Sharp et al., 1995). Children can easily express their thinking about the story using verbal articulation but struggle to use a sequence of details. Some children may include each aspect of the story as they remember, where others may jump around. Still, others may be reticent to share what they know, as their executive skills for formulating the story are still developing. Therefore, an adult can facilitate the child's retelling by managing the meta-skills needed to decompose a story and identify its structure. They may prompt children with questions, for example, who is the story about? What did they do? Where were they? Who were they with? What important events happened first? What happened in the end? These literacy skills help a reader break down, or decompose, the story into its elements. With an adult's assistance, a child can explore the structure of stories (literature) and how to structure the sequence of events, actions, and narratives of the characters in the story. Therefore, in many ways, the construction of a story parallels to the design of systems.

Changing the medium of a story, particularly animation tools, provides abundant opportunities for young children to retell their version of the story. Literacy seems a natural analog for computation, and many speculate the same skills children learn in literacy might transfer to programming. Thompson and Tanimoto (2011) performed a literature review in which they generate a framework for literacy in coding by blending literacy literature with programming skills. While they did not test these ideas, they define criteria for successful programming/literacy hybrid:

> *"fosters programming and narrative skills simultaneously; makes programming feel as natural at an early age as storytelling; is accessible to each child regardless of individual skill level in programming or narrative; and accommodates any child's reading ability, from illiterate to fluent."*
> *(Thompson & Tanimoto, 2011, p. 7)*

Thompson and Tanimoto set a high bar for a programming environment as by its nature programming tend to be exact and demand a level of competency. Several technologies have been created however to lower that bar, two of which, ScratchJr and Alice, have been studied extensively. These authoring environments provide learners with resources they can use to express their ideas by designing and building animations that tell stories. With the resources in ScratchJr, children can choose scenes and characters. They can write programs to autonomously move characters about the scene and express their thoughts with either with dialog bubbles or recorded audio. Scenes can connect to create a story flow through various settings and actions.

The Alice framework allows users to create 3D animations which can be controlled to tell stories. While stories are traditionally shared verbally or through writing, introducing the new media was generative for users as "the library of 3D characters with custom animations… inspired stories" (Kelleher & Pausch, 2007, p. 61). Users of Alice created elements of their stories by looking at the capabilities Alice provides which inspired their storytelling. This phenomenon highlights an essential interaction between storytelling and the chosen media. Is it reasonable to teach CT devoid of the capabilities of the technology? If the technology is needed, how fluent must individuals be in the technology to make the best use in problems solving? How much do the available tools influence the problem-solving?

This paper explores how ScratchJr as a medium influences storytelling (retelling). We introduced first-graders to ScratchJr by having them retell stories through animation. We compared their initial plans for telling the story to their initial design and then finally the resulting animation. We assess how the stories children tell evolve as the participants learn more about the capabilities of ScratchJr and look to see how the results of general problem-solving in literacy change when then created in a specific technology medium. This comparison may shed light on how 'technology neutral' CT can be, versus how much technology influences problem-solving.

Methods

Participants and Context

This study involved one first grade class in a Title 1 school located in the Midwest. The selected classroom of 18 participants had recently completed an integrated STEM, literacy, and computational thinking unit as part of our NSF STEM+C grant. The class engaged in two half-day computational thinking lessons developed for use in a first-grade classroom. The first half-day lesson included seven activities using the Learning Resource's Code & Go Robot Mouse Activity set, exposing participants to creating sequential only algorithms using a programmable hardware device. The second day expanded to introduce ScratchJr allowing for more robust programs and open-ended storytelling animations. The content for the second day is shown in Table 1, though the focus of this paper is on the final activity on day 2, retelling a fairy tale. The initial activity contained three projects designed to train students on how to use the basic features of ScratchJr. Each project gave students the opportunity to script a short scene. Scene construction consisted of selecting a backdrop and characters, generating movement for the characters and their dialog (text or audio recordings).

Table 1. ScratchJr Lessons from Day 2

Activity	Description
Learn about ScratchJr	The teacher provided a basic orientation to the iPad app. Then students explored the visual programming blocks. Students then shared with the class what they discovered about the buttons.
Mini Project 1: Race and Celebrate	Students made three characters race across the screen. The final product had to contain a background and three characters. One of the characters must finish first and then celebrate their win. Students learned how to write algorithms to control the motion of multiple characters and how to utilize a variety of actions to tell a story.
Mini Project 2: Tell a Joke	Students made a character tell a joke to another character. Students practiced using speech bubbles and their recorded voice to communicate a story.

Project: Retell a Fairy Tale	Students completed the final project in teams of three. Teams selected a fairy tale, and each member decided if they were going to retell the beginning, middle, or end of the story. Students used individual story planning sheets to identify the main characters, setting, and significant actions that should take place in their retelling. They then used a code planning sheet to communicate which buttons they would use to make their planned actions happen in ScratchJr.

Data Collection and Analysis

For this paper we are focusing on the last of the projects, retelling a fairy tale. We chose this since the children have already had at least a basic introduction to ScratchJr, and it allows for the most in-depth collaborative creation. Students completed two types of planning sheets, one for breaking down the story and another for documenting their ScratchJr code. We collected planning sheets and analyzed the student's intent in storytelling before turning to their animation. Each student was provided an iPad with ScratchJr to work on independently. Their final animations were captured on video. Additional iPad captured video from each of the tables where the six groups worked to see their progress and interactions with each other and researchers. In total, we were able to collect data from 8 participants including their storyboard, code plan, and final code which is used in the comparative analysis described below.

Since the goal of this research is to see any developing patterns we chose a qualitative open coding scheme. We used the participant storyboard to capture how participants broke down the story and the story they planned to tell. The code worksheet provides the first evidence of how they would apply the ScratchJr features they had learned to the story. The final product would demonstrate the actual story told, as a comparison with the original plan. Since this is the third project in which students are using ScratchJr, they should have a basic understanding of the capabilities of the tool. The prior projects allowed some experience in the capabilities of ScractchJr animation, and in theory, their storyboard should accommodate ScratchJr. For most, or all, participants, this is their first experience with ScratchJr, so they likely need introduction to the features of the ScratchJr app. Thus, participants have exposure to ScratchJr but have yet to adopt ScratchJr's capabilities into their way of thinking. By comparing the three phases of their project, we hope to infer how thinking shifts from literacy to animation via computing. The discussion following may refer to numbers, but these are intended to describe the prevalence of phenomena, not to imply statistical meaning.

Results and Discussion

Students seemed to enjoy using ScratchJr for storytelling genuinely. In each of the six videos, students are focused and work the entire time, which is not always the case with such demanding tasks and the age group. Students did not need to be prompted to stay on task. Researchers redirected students who spent too much time on less important tasks like redrawing characters in the paint program. Generally, students were able to freely choose which technology they included and how they formulated the story, even if that meant ignoring earlier plans. The final work for the participants varied greatly in complexity and quality. Some of the animations were limited to moving a single character, while others involved interactive motion, recordings, and dialog among many characters. Regardless of the final complexity, students appeared engaged through the entire process. Each participant succeeded to some degree or another with a few creating quite robust animations. The activity, as an introduction to design and programming within a literacy context, was successful. Whether the activity created lasting skills and noticeable progress in specific concepts is difficult to demonstrate. Our goal is to evaluate how plans changed (or not) from storyboard to animation, and we saw an interesting spectrum even in our limited dataset.

Storyboarding and Literacy Skills

The participants' literacy skills seemed to represent a broad spectrum of interest and/or ability. While literacy is an actively covered topic in elementary education, not all students will come to this activity with the same proficiency. Only half of the students seem to be able to break the story into a beginning, middle, and end, clearly identify which part of the story they are attempting to retell or tell the story in any detail. Nearly all participants can describe the action going on in the story, but few include any dialog or overall description of the scene they are planning to create. The worksheet used to capture the storyboard plan may be a confounding effort in this evaluation, as it may cater to students who are stronger or more patient writers. At one point in the video, a researcher approached a participant to check in on her progress. She showed the animation but told an elaborate story about how a man is selling sticks to one of the pigs building their house, which was nowhere near the animation. It is possible the medium is not a strong representation of the ability to retell or even analyze a story, but, as shown in Figure 1, the worksheet still allows for limited descriptions and provides a nominal structure for information.

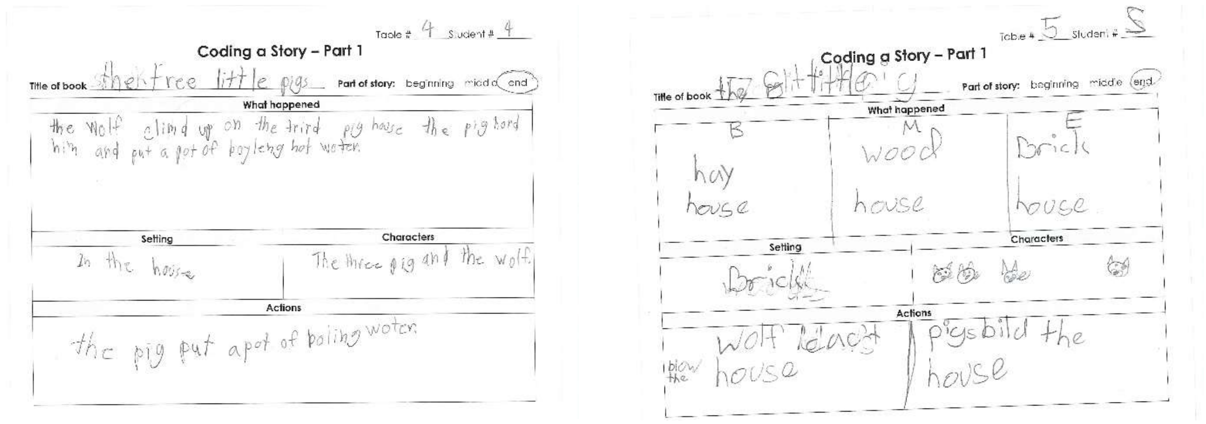
Table # 4 Student # 4

Coding a Story – Part 1

Title of book ~~s~~thek free little pigs Part of story: beginning middle [✓ end]

What happened

the wolf climd up on the trird pig house the pig hord him and put a pot of boyleng hot woter.

Setting	Characters
In the house	The three pig and the wolf.

Actions

the pig put apot of boling woter

Table # 5 Student # S

Coding a Story – Part 1

Title of book the ~~B~~lit tittle p'g Part of story: beginning middle [✓ end]

What happened

B	M	E
hay house	wood house	Brick house

Setting	Characters
Brick	

Actions

blow the wolf [illegible] house	pigs bild the house

Figure 1. An example of stronger (left) and less apt (right) storyboard worksheets.

The examples in Figure 1 demonstrate two ends of the spectrum of storyboard analysis. The participant worksheet on the left provides a description of the action, the setting, characters and main action in their scene. The right worksheet is retelling the entire story, only capturing the main objects, not actions or descriptions. The action sequence is a rather high-level abstraction, (build a house, blow it down) and notably out of order for the story. While these two are both telling the rather simple fairy tale, they clearly are at different developmental levels of being able to break down and describe the action. Their capabilities may also impact their ability to follow through with their portion of the story in animation.

Planning and 'Design'

The design task required participants to move from their storyboard into a plan for the scene they would construct. Moving from a storybook retelling to an animation required the participant to move from abstract actions (e.g. "blow the house down") into physical motions that are highly limited to the offerings of StratchJr (e.g., move in 4 directions, rotate, disappear).

The participants were instructed on three main action features of Scratch Jr, character movement, creating dialog in 'speech bubbles', and playing recording audio, as well as basic controls such as appearing/disappearing and pausing, but not on all the capabilities ScratchJr provides. Most participants planned code did not reflect the storyboard they had constructed. Only one participant stuck to the planned story from start to finish, with two others keeping elements of the same story, but most animating some other aspect entirely. Watching students work on their design, they never seem to revisit the storyboard. Instead, their storyboard often sits right under their tablet or to the side of the design paper ignored. While it may be useful to remind students of their original plan, this was not part of the protocol and showed that the storyboarding process does not seem to naturally influence the design task the participants are asked to complete.

The design plans completed by participants neither show a reflection of the storyboard nor an understanding of ScratchJr's capabilities. The participants used motion commands the most in design, though a few used 'speech bubbles' and perhaps a few may have used a symbol indicating recorded playback. All but one design included some basic movement of characters (the last involved no instructions, the final project being recorded dialog only). Some participants define simple motion, but in other cases motion is extended beyond the demands of the storyboard, and even beyond the final implementation in code. For example, the participant who created the left storyboard in Figure 1 created the left design in Figure 2, which portrays a simple motion that may or may relate to the boiling pot of water. The right side of Figure 2 shows the original storyboard (at the top right) which plans for the dragon to throw fireworks, but includes actions for five characters and no fireworks! This participant was either confused, chose to change directions, or saw these as unrelated tasks, yet the final project created a dragon throwing fireworks. It includes a dragon and a rocket (plus a second dragon?), so middle worksheet seems unrelated to the storyboard and final product.

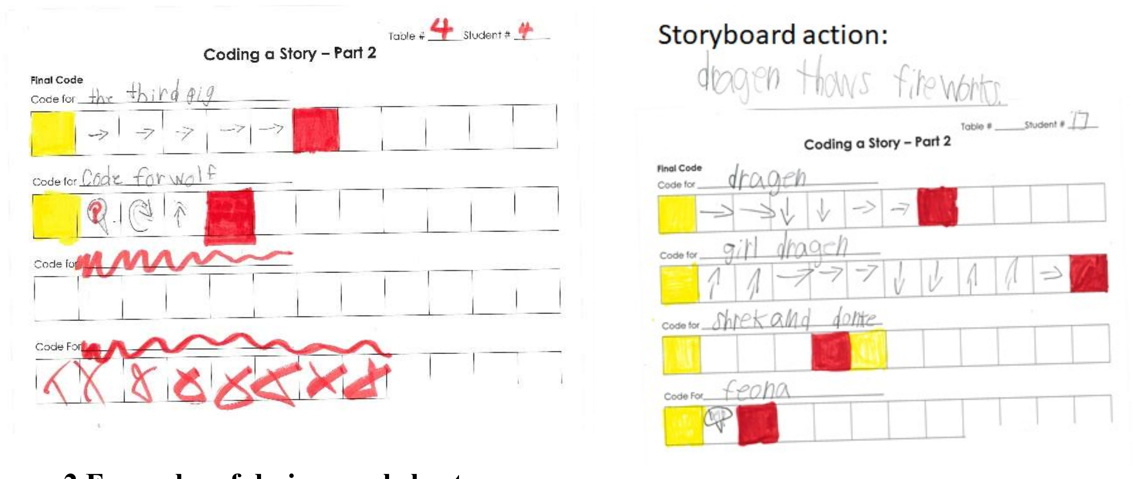
Table # 4 Student # 4

Coding a Story – Part 2

Final Code

Code for the third pig

Code for Code for wolf

Code for

Code For

Storyboard action:

dragen thows fireworks

Table # ____ Student # 17

Coding a Story – Part 2

Final Code

Code for dragen

Code for girl dragen

Code for shrek and donke

Code For feona

Figure 2 Examples of design worksheets

The 'code design' worksheets are a bit of an anomaly in the process and data. The worksheets did not seem to be a critical step in building the animation. Some students constructed much of their code in ScratchJr and came back to fill in the worksheet. These participants did not copy their final code to the worksheets, however. One participant can be seen working with ScratchJr and taking a 4-minute break to go fill out the design worksheet yet during this time he only looks at the tablet briefly. He did not refer back and forth between the table and paper enough to be transliterating code, as is proven when comparing the final work and the final animation. It is difficult to determine the role the 'design' worksheets may or may not have played in participant thinking. They do not appear to be integral to their translation from storyboard to animation, and this case may indicate they are a non-sequitur.

Creating Animations

The final student projects show a lot of growth/deviance from the plans. Every student completed some working animation, though a few were less creative/advanced than others. About half of the final projects have some resemblance to the design code, with only a third showing resemblance to the original storyboard action. The ones that matched the best were the most straightforward actions. The storyboards with abstract behaviors were the toughest to translate (e.g., build the house). These participants tended to modify the original plan into linear character motion instead. The simplification of animation would imply that when ScratchJr did not provide a natural analog, the students chose to simplify their original plan. It may be that the participants with the simpler plans had simply adopted quicker to the limited toolset within ScratchJr, while the more ambitious storytellers started with grand plans but settled due to the time constraints and limited features in Scratch Jr. In either case, the final animations tended to be more straightforward than the rich vocabulary of the original plan in storytelling.

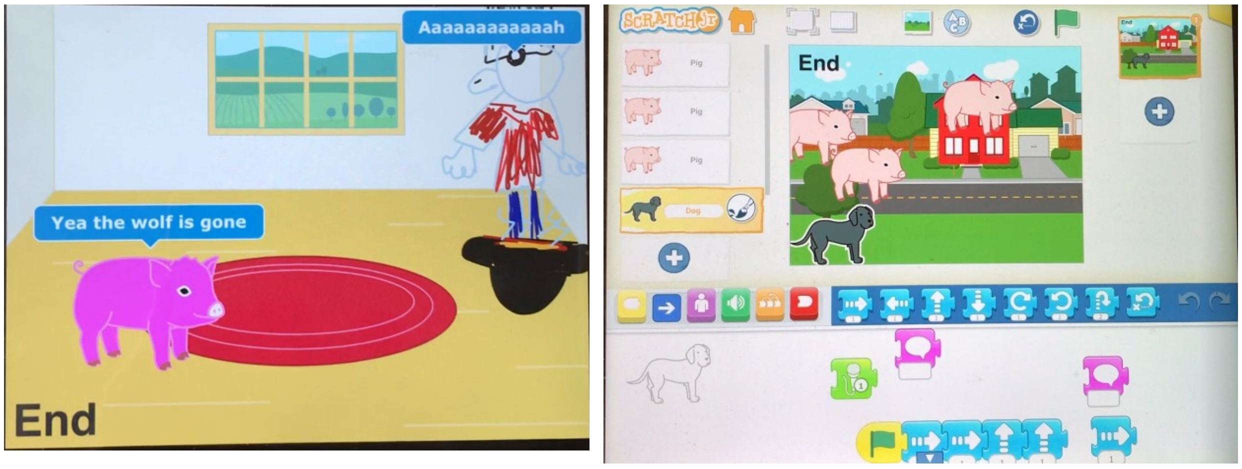

Figure 3 Final result from the same students as in Figure 1.

A simple initial plan did not mean that the final product was simple. If the plan called for just simple motion, like our 'boiling water' participant above, they might add additional dialog or play with the custom character creation as shown on the left side of Figure 3. While students with simpler initial strategies seem to spend more time in exploration, it is not to say the final product was inferior due to a 'lesser' initial storyboard breakdown. Some students chose to compensate for the limited animation options by incorporating narrative storytelling using dialog (written or recorded). The participant on the right side of Figure 1 had a "weaker" storyboard but created a very clever animation (right side of Figure 3) that animated the three pigs racing the wolf to the house and disappearing inside. They were able to remove the images of the pig in perfect timing with their arrival at the house. This screenshot of their work shows where they tested the recorded audio and dialog capabilities, though unconnected in the final product. The simpler storyboard did not seem to indicate that the participant would struggle in the coding phase.

Tying CT and Literacy

Participant performance in retelling stories varied, yet it seems clear that growing knowledge of the capabilities of ScratchJr modified their use of the medium. One participant (the left pictures in Figure 1, Figure 2, and Figure 3) started very simple but told a much more complicated story as they better understood the ability to add dialog to their scene. Another participant had a more abstract idea (the right pictures in Figure 1 and Figure 3) and modified their animation into something more straightforward, yet still telling a crucial part of the story. As students learned more about ScratchJr, they seemed to want to test out the new capabilities and use them in their projects. For some students this meant expanding their original idea, for others, it led to a different direction altogether. What seemed consistent, they did not associate abstract planning with the final work. Participants created the storyboard worksheet and forgot it. Creating this worksheet may have helped structure the thinking that came later, but participants neither referred to them while coding nor were they a reliable indicator of the final product's content in our participants. It is difficult to judge how much of the final animation came from careful planning and how much was the result of tinkering until time ran out. We saw cases where students were able to align their planned design to their code, and some students tested out several code blocks that did not make it to their final product (e.g., right side of Figure 3). Given the nature of a two-day intervention on CT and programming, the outcome is unlikely to be a lasting understanding of CT, but it seems clear that ScratchJr is a powerful accelerator for understanding computation than abstract storytelling alone.

Conclusion

Computational Thinking seems to be most valuable in young learners when it is grounded in concrete activities. Perhaps if the class already had extensive ScratchJr experience, they could have better spent time in abstract planning for storytelling. How engaging would it be to plan out stories without the payback of seeing your final animation? We saw students at various levels of competency in the literacy aspects of storytelling, so adding a goal of CT on top of this could diffuse learning in both. One of the limitations of our approach was a lack of time, staff, and perhaps foresight to consider testing. On reflection, a powerful lesson could have been made by stringing together the three stories from the groups to see if they aligned. We could see some evidence of groups working together, while other groups worked independently. This final step would have reinforced both the literacy concept of beginning, middle, and end as well as the CT concept of decomposition and algorithm design.

ScratchJr provides a powerful tool for open-ended play that relates to both programming and literacy if used correctly. Using a tool successfully in one context does not mean developing lasting competencies that will transfer to other tasks. It seems an essential part of defining computational thinking is not just finding analogs for activities such as decomposition and defining algorithms, but to ground those activities in the capabilities of the chosen computational technology to solve the problem. It seems unlikely that the ScratchJr and Robot Mouse experience would make learners hit the ground running on other tasks. While specific ideas may provide confidence, they are a far reach from the rigorous demands of most computational systems. It seems absurd to assume first graders who have coded in a block language to progress to more complex programming languages, but it is not that different than what educators assume of older learners? Pea and Kurland (1984) once wrongfully speculated that learning to code would help problem solve in other areas, so it seems unlikely that anything less than active CT activities within a specific technology will develop lasting skills. ScratchJr seemed to be fun and motivating for our participants but would only be an early step on a longer path to being computational thinkers.

It does appear that the computational medium has a significant impact on story retelling. Only one in three animations remained faithful to their original storyboard, and these were the simplest. Either these third of students already understood that ScratchJr would only allow for simple animations, and created their story accordingly, or they were not rich verbal storytellers, to begin with. For the students who preferred rich storyboard creations, they were forced to tell simpler stories or fall back on dialog to tell the depth of their original story, merely sequencing the text rather than genuinely changing to a new medium. It may seem logical that problem-solving is bound to the tools available to solve the problem, but since "computers can do anything" it may bear noting. In the right hands and with enough time a computer can do most anything, but the capabilities exposed by a simpler toolset limits the creative potential for novice programmers. It seems that understanding the computational medium (ScratchJr) at the appropriate level of abstraction (the available commands) provides the path of least resistance to building the desired animation (constructing a storyboard which can be seamlessly animated).

ScratchJr provides informal learning experiences that engage children's imagination and literacy skills to design stories they can share with others. Building stories with ScratchJr require planning and literacy skills which are enriched as learners transform their ideas into an animation. Working with the iPad application, children also become comfortable using technology. Future work will continue to refine the interaction between computational storytelling and literacy development and how teachers can foster it within formal learning settings.

References

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Annual American Educational Research Association Meeting, Vancouver, BC, Canada*, 1–25. Retrieved from http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Common Core State Standards Initiative. (2010). National governors association center for best practices. *Washington, DC: Council of Chief State School Officers*.

Computational thinking concepts guide. (n.d.). Retrieved November 2, 2016, from https://docs.google.com/document/d/1i0wg-BMG3TdwsShAyH_0Z1xpFnpVcMvpYJceHGWex_c/edit

Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Kelleher, C., & Pausch, R. (2007). Using storytelling to motivate programming. *Communications of the ACM*, *50*(7), 58. https://doi.org/10.1145/1272516.1272540

Lowe, T., & Brophy, S. (2017). An operationalized model for defining computational thinking. In *Frontiers in Education*. Indianapolis, IN.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, *2*(2), 137–168. https://doi.org/10.1016/0732-118X(84)90018-7

Selby, C. (2013). Computational Thinking : The Developing Definition. *ITiCSE Conference 2013*, 5–8.

Sharp, D. L. M., Bransford, J. D., Goldman, S. R., Risko, V. J., Kinzer, C. K., & Vye, N. J. (1995). Dynamic visual support for story comprehension and mental model building by young, at-risk children. *Educational Technology Research and Development*, *43*(4), 25–42.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Thompson, R., & Tanimoto, S. (2011). Children ' s Storytelling and Coding : Literature Review and Future Potential, (1997), 204–212.