Children's Storytelling and Coding: Literature Review and Future Potential

Rob Thompson
Computer Science and Engineering
University of Washington
robthomp@cs.washington.edu

Steve Tanimoto
Computer Science and Engineering
University of Washington
tanimoto@cs.washington.edu

Abstract

Children are now learning programming as early as in primary school or even earlier. The various programming constructs and patterns of use involved in coding require different levels of cognitive development, and young children are not ready to tackle all levels yet. It is well known that some aspects of programming are accessible to young children. We suggest that even more aspects may be accessible through the use of storytelling. In the course of this discussion, we survey literature relevant to the question of how storytelling and teaching programming can be mutually supporting. Then we describe relationships between six forms of programming and five levels of narrative development. Finally, we discuss three prominent issues regarding children's coding and storytelling.

1. Introduction

Computer science education is expanding in primary and secondary schools in many countries, such as the US and the UK. There is a great deal of interest in having more children learn computer science earlier for a variety of reasons. One reason is that it promises to be a subject that will see a lot of job growth in the coming decades, and exposing children at an earlier age makes them comfortable with key topics and more likely to pursue the subject as adults. Another reason is that coding is an avenue towards computational literacy (Wing 2006).

There is some debate as to what age is appropriate for beginning computer science instruction. Duncan et al (2014) argued that multiple factors should contribute to the decision, including what tools are to be used and whether knowledgeable teachers are available. Young children may simply not be at a stage of development that allows them to grasp some computing concepts. However, young children can understand some basic concepts of computing, and these concepts have parallels with storytelling. In this paper, we examine the obstacles that prevent young children from understanding key concepts and then discuss several projects that combined storytelling and basic programming at early ages. Then, we examine the stages of cognitive development related to narration and how they may support teaching basic programming concepts. Finally, we consider three related issues: how school curricula should incorporate children's coding, the incorporation of game elements into storytelling and coding, and criteria for an ideal programming environment for children to use in storytelling.

2. Childrens' Limited Powers of Abstraction

In Piaget's theory of cognitive development, children reach different levels of abstract reasoning ability as they age (Piaget and Inhelder 1969). Children begin life in the sensorimotor phase and pass through the preoperational and concrete operational reasoning phases before arriving at the formal operational reasoning phase, which starts at approximately eleven years of age.

Lister (2011) mapped these Piagetian stages to programming abilities and stated that full programming ability is not present until the final stage of development, the formal operational stage. Specifically, Lister applied these stages to Neo-Piagetian theory, which states that individuals go through the four stages of development numerous times, regardless of age, as they encounter new problem domains. While any individual can have trouble learning to program at any age, this theory suggests that typically developing children will not be able to fully program before the age of eleven.

This notion is supported in programming education research as well. Rader, Brand and Lewis (1997) tested children in grades 2 to 3 and 4 to 5 on their ability to program by demonstration in the Cocoa language. The complete list of tasks included individual actions, rule order comprehension, picture matching, object interaction, subroutines, and properties. The majority of 2nd and 3rd graders could not complete anything past the individual actions, while the 4th and 5th graders understood all but properties. Our interest is in children younger than 2nd graders, but if older children are unable to grasp these concepts, younger children will almost certainly be unable to grasp them too.

While full programming may be out of reach for young children, researchers have still devised and tested programming environments for them that use some aspects of programming. Morgado (2005) discusses these systems in great detail in his thesis work. We will highlight only some of them below.

TORTIS is one of the earliest examples, developed during 1974-1976. Radia Perlman created TORTIS as a tangible programming system that grew out of the LOGO project (Morgado 2006). LOGO itself was a children's programming language by Papert and others but required some ability to read (Papert and Solomon 1971; Papert 1980). In TORTIS, children built programs out of physical blocks that would then control a robot that could move forward, backward or rotate, lower and raise a pen, toggle a light, and beep. The TORTIS environment supported command repetition and up to four user-defined functions. Because TORTIS was entirely icon-based, children as young as three used and enjoyed the system.

Similar to TORTIS, Microworlds JR is an icon-based implementation of LOGO (LCSI 2004). Children construct programs by clicking and dragging icons onto a program track. Through commands, children control an on-screen turtle, and can instruct it to move, draw, stamp its shape, play music, and more.

In Cocoa, TORTIS, and Microworlds JR, students control character movement directly or through commands. Both are limited in their ability to support storytelling, however. Other systems, such as Alice (Conway et al. 1994), or Kokopelli's World, a system we ourselves are developing (Thompson at al 2016), more explicitly support storytelling, such as with dialog support and custom animations. Before we discuss storytelling environments, however, let us look at the value of storytelling.

3. The Value of Storytelling

Children enjoy stories from a very early age, and storytelling as a form of instruction has many benefits. People pay more attention to stories than other text and are better able to remember information in the form of a story (Willingham 2004). The power of storytelling has already been featured in several computer science education research projects. Kelleher and Pausch (2007) used storytelling to motivate girls to learn coding in the Alice environment. Howland and Good (2015) explored the use of storytelling and natural language in programming in their work with Neverwinter Nights and the FLIP language.

Older computer-based storytelling environments for children include Spinnaker Software's Story Machine (InfoWorld 1983) which immediately animated story events typed in English on the keyboard, Programming by Rehearsal, which combined a drama metaphor with graphics and the SmallTalk language (Gould and Finzer, 1984), and PLAY, which provided an iconic scripting facility with an animation facility using the drama metaphor (Tanimoto and Runyan, 1986).

A few research projects have both targeted young children and used storytelling as a vehicle for programming.

4. Young-Age Storytelling Programming Environments

Scratch Jr. is an attempt to adapt the blocks-based Scratch programming environment to K-2 students (Flannery et al 2013; Resnik et al 2009). It focuses primarily on story creation. Scratch Jr.'s design goals were to offer an environment with a low floor and a high ceiling that can engage students at a young age but still support creative solutions as more complex languages do. Programming in Scratch Jr., as in Scratch, involves connecting blocks together to form sequential commands. Blocks' labels are icons, to enable pre-literate users to interact with the system. Most commands involve the manipulation of one of several on-screen 2D sprites, and nearly all have immediately visible effects. These sprites represent the characters children use to tell their stories. Stories/programs can also be shared between students, and Scratch Jr. offers classroom support for teachers to facilitate multiple students using the language simultaneously.

ToonTalk is an earlier example of a system designed for young children (Kahn 1996). ToonTalk provides a constraint-based system and presents itself like an open-ended Lego sandbox. Children program agents by adding blocks, all represented by icons instead of text, to a work space and programming by demonstration. While ToonTalk could in principle be used to tell stories, it presents itself more as a game where users can manage a town and create their own games to share with friends.

Magic Forest is another icon-based programming environment that allows users to edit or create new games by placing "stones" (commands) inside of "scrolls" that are attached to objects (Andrade 2007). The scrolls dictate the object's behaviour. Like ToonTalk, Magic Forest presents itself as a game making and playing environment more than a storytelling platform. Storytelling is an integral part of game-making, however, and tools to build games can also be used to craft stories.

The role of animated graphics in storytelling environments is important both as a means of semantic feedback to children and as a source of humour and richness. This could be seen even within the relatively primitive environment of Spinnaker Software's Story Machine (InfoWorld 1983). With more modern graphical affordances such as those described by Pahud (2016), one can expect children's motivation to be stronger.

Storytelling has been used by these systems primarily to contextualize and motivate programming actions. From a young child's perspective, storytelling and programming may be quite similar tasks. There are separate theories on childhood narrative development, however, and thus there may be something to be gained from using these theories to re-examine how we present programming tasks to children.

5. Narrative Development Levels

Stadler and Ward (2005) presented five levels of narrative development they observed while listening to stories told by 3-to-5 year old preschool children: labelling, listing, connecting, sequencing, and narrating.

In the labelling stage, children primarily talk about objects or character that they clearly have at their disposal. There are few actions and no expression of story flow.

> No, that's not my cat. That's my cat. That's her cat. This is, and this is bee. Here's my girl.

In the above example, the cats, bee, and girl are mentioned but not related to each other. Neither do any of the characters perform any actions.

In the listing phase children's stories typically resemble long lists, but start to include attributes and actions of objects.

> My picture is a XX. And it have, and it has kids with music. And there's some guy who's teaching them how to do music. And then trying to make it. Some of 'em are not listening cuz that one's who's being, like (gestures) are doing that. This one's doing that. And so he broke the wire with the call the phone. (claps) He break it and the guy's drinking some soda. And they're doing their music concert. And the end.

Here, connecting words appear, such as "and" and pronouns (it). Verbs are being used as well, like "teaching", "broke", and "drink".

In the third stage, connecting, objects and actors more commonly interact with one another, though there is still little to no sense of temporality until the fourth stage, sequencing.

> I have a garden by my house. And, it, um, I have a dog. And my dad puts her poop in the garden. Yeah, because that's the only place we can put it. So he puts it in the garden. And we have some little pink flowers growing in there. And, um, they, um, my grandma and gramp came over. And they were going to check one day. And then we saw those red flowers and they were blooming. And, um, um, my mom always

> goes to the garden. And she takes a watering can and waters them so they grow. They grow, but not too often in the spring.

In the connecting phase, verbs more commonly have subjects and objects, like “my dad puts her poop in the garden.”

Sequencing sees the introduction of stories that exhibit temporal ordering or cause and effect relationships.

> On my birthday, I was holding my cat. And then my Mama took a picture with my brother holding it. And I was holding his head. And it was Jessica, my big sister’s cat. And her name is Callie. But she doesn’t have front nails. And she’s very little, because Cindy took her to the doctor. And then the doctor cut all her nails out. But it didn’t hurt at all. She couldn’t feel a single thing.

Here, the child is able to stay focused on one topic for the whole story, and the last three sentences are each a result of the preceding sentence.

In the last stage, narrating, children begin planning their stories on higher levels, and incorporate relationships such as foreshadowing. At this level, children have to understand what their story is going to say from beginning to end.

> As she looked up, she saw her fairy godmother. And the fairy godmother said, ‘‘No wonder you’re so sad. I must make you a coach.’’ And she did. And Cinderella said, ‘‘Don’t you think my dress?’’ ‘‘It’s wonderful!’’ her godmother said. And she looked again. ‘‘Oh, good heavens, my child, you couldn’t go in that.’’ So Bibbety, Bobbety Boo. There stood Cinderella in the most perfect gown. And Cinderella said, ‘‘This is wonderful. It’s like a dream.’’ And the prince danced with the charming Cinderella. And the king said, ‘‘That prince danced with that girl all night. So I think that means he found the girl that he wanted to marry.’’

In this example, the child had a clear notion of how the story would conclude from the start, and events flow logically and consistently.

If children conceive of programs and stories in the same way or similar ways, then there may be value in teaching programming concepts in a way that mirrors the levels of narrative development. Here we note that Stadler and Ward are not the only ones who have presented theories of cognitive narrative development. Their model could be considered an extension of earlier work done by others, including Applebee (1978) and Vygotsky (1962). Vygotsky’s stages in particular have several similarities to Stadler and Ward’s stages. However, to avoid clouding the discussion, we only examine one of these models in detail, and choose to focus on Stadler and Ward’s as it is the most modern. The next section adopts a definition of programming and shows how forms of programming could map to these levels of narrative development.

6. Relating Programming to Cognitive Narrative Development

Blackwell defines programming in broad terms in “What is Programming?” (2014). While Blackwell’s definition is by no means the only one, his definition attempts to address more varied and modern forms of programming, such as end-user programming, while still remaining broadly applicable. He describes programming as composed of five types of activities: requirements, specification, design, coding, and debugging. As mentioned by two of our reviewers, Blackwell’s definition of programming does not specifically include one type of activity particularly common among young programmers: recoding, also referred to as reuse or remixing. Dasgupta et al define remixing as “the reworking and combining of existing artefact’s” (2016).

Table 1 shows a matrix of narrative development levels (with one for each column) and programming activities (with one for each row). We use the letter *x* to mark potentially interesting relationships, while letters *a* through *h* mark relationships we feel are supported by prior studies.

	Labelling	**Listing**	**Connecting**	**Sequencing**	**Narrating**
Requirements	a	x			b
Specification		x	x	c	
Design				d	e
Coding	f	x	x	x	x
Debugging	x	x		x	g
Reuse	x	h			x

Table 1 - Relationships between cognitive narrative development levels and programming activities.

Program Requirements

Blackwell defines the requirements component of programming as "decid[ing] the intended result of executing the program." This includes specifying the overall purpose of a program, along with its key stakeholders and gross functionality. Similar steps occur when creating a story, even at low development levels like labelling or listing. At these stages, children are still clear about who or what is in the story, and what the story will literally be about; for example, "This is a story about my cat." Requirements rarely involve the "how" and as such are a good fit for early narrative stages, which rarely involve actions or intent.

(a) For Flannery et al in their paper on Scratch Jr. a habit they commonly observed in both Scratch and Scratch Jr. is for children to create lots of characters with no scripted behaviour (2013). This occurred most often with younger, less experienced students who were lost or not engaged. This sort of behaviour is similar to what one would see in the labelling or listing stage of narrative; creating numerous characters or nouns with little attention spent on actual actions.

(b) There is also a higher-level component to requirements: One has to consider one's audience. Knowing what an audience wants requires a degree of empathy that younger children may not be capable of until later stages of development, such as the narrating stage. A study of 4th graders using the LaPlaya blocks-based environment found that they rarely created programs for anyone other than themselves (Hansen et al 2015). The study did not prompt them to make programs for others, so it may be that they are capable of doing so, but it does not appear to be something they do instinctively even by the 4th grade.

Program Specification

Specification involves "identifying when [the program] will be executed, and allowing for variation in different circumstances." At the specification stage one must decide what form the program's input and output will take, as well as the overall relationship between that input and output. Context becomes more important as well. In the narrative levels, actions start to appear in the listing and connecting phases, and cause-and-effect relationships appear in the sequencing stage. The relation between inputs and outputs cannot be properly understood before familiarity with the cause-and-effect concept.

(c) Rader, Brand and Lewis, when observing the 2nd and 3rd graders who were unable to use Cocoa to perform complex commands, found the children could describe high-level ideas but were unable to break those down into a series of actual actions the system could recognize. Neither could the children specify the correct order for a series of actions. These students may have been succeeding at the requirements level but failing at the specification level. Since they also could not sequence commands correctly, the sequencing narrative-development stage may be an upper limit for specification.

Program Design

The design stage is the one in which the programmer "chooses from a set of technical features that may support [the desired] behaviour." At this stage "how" becomes the central question. Choosing an effective design strategy requires one to be familiar with multiple strategies already, and be able to evaluate which is most effective. From the narrative perspective, this requires one to be aware of the relationship between actions (sequencing) and to have a clear notion of the purpose of the story (narrating).

(d, e) Herbert Simon (1969) popularized the notion of design as a complex, scientific process. We can consider design as a form of complex problem solving. Design has a very high skill ceiling that can accommodate an unlimited degree of advanced reasoning. As such, design activities most likely map to higher levels of narrative development as well.

Coding

Coding can be as straightforward as "entering abstract control commands as well as data." But as described in the previous sections, coding can involve a wide range of abstract reasoning capabilities. We argue that actions involved with coding can map to any stage of the narrative development process.

(f) A study examining the habits of children using Scratch found that children greatly prefer a bottom-up programming approach, where they immediately started executing commands and then edited them until they got the desired behaviour (Meerbaum-Salanit et al 2011). Experience with this and other systems already discussed, such as TORTIS, indicates that children are capable of performing coding tasks at a low level of abstraction without much planning. Coding has a high ceiling, however, so particular tasks map to the narrative development levels depending on their degrees of complexity.

Debugging

In debugging one must "anticipate and account for departures from the intended behaviour." Because one must already have a program, and have a clear sense of what the intended behaviour is, debugging is a fairly high-level activity. As such, we feel it falls more in line with sequencing and narrating narrative stages. A notion of cause and effect is essential to discovering what is causing a bug, and fixing a bug requires a complete understanding of one's program or story to avoid adding new errors in the process.

This is an idealized vision of debugging behaviour, however. In practice, beginning programmers, and especially children, may be able to identify bugs without being able to reason through them yet. Children may blindly change code in an attempt to resolve an issue they do not understand, or follow coding standards told to them but which they do not yet fully appreciate. While these sorts of debugging methods should not be encouraged, they are more likely to map to earlier stages of narrative development.

(g) A survey of UK Code Clubs, with over 2200 children responding in total, found that children were least confident in their ability to debug code over general programming, variables, conditionals, Booleans, and broadcasting (Smith 2014). These were children who had completed an average of six Scratch projects. That children at this level were the least confident about their ability to debug suggests it is at least perceived as one of the more advanced types of programming behaviour.

Reuse

Retelling is a fundamental aspect of all storytelling, and as such reuse can appear in many forms throughout narrative development.

(h) Naïve forms of reuse can be seen at early stages of programming and narrative. In the labelling and listing stages, children seldom create new characters, but rather include entities they know from their own lives or from other stories they have heard. Even in later stages of narrative development, children are copying techniques and themes they have seen elsewhere before (Hill and Monroy-Hernandez 2013).

7. Discussion

The matrix above highlights the many potential relationships between narrative phases and forms of computer programming. Now, we raise three broader questions that arise as consequences of attempting to integrate storytelling and the teaching of programming.

As educators and technologists attempt to empower children to program at earlier ages, an important question is how school curricula should change to accommodate this. As we have attempted to show, there are potential synergies between storytelling (and therefore language arts more generally) and coding. Arguments can also be made for creating synergies between coding and other subjects: mathematics, science, art, music, and physical education. By focusing first on achieving an integration of coding and storytelling, we may be able to begin to design new curricula at the beginning of the child's formal education and work up from there.

How does interactivity fit into storytelling? Game-creation has been a popular approach to computer science education in recent years, and often students would rather create story-like experiences with interactive possibilities than traditional single story-line narratives. What are effective ways to combine the benefits of storytelling and game-making in the context of computing education?

What would an ideal programming environment for storytelling look like? Perhaps there is no single environment that could be both general enough and tailored to the special needs of storytelling that it could be called ideal. However, as we've suggested in this paper, designers ought at least to consider a theory of child development in terms of narrative, and work out how children could use their system effectively at various development stages. Ideally, such a system would meet the following criteria: 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.

8. Conclusion

We have discussed some of the issues that prevent young children from becoming full programmers, as well as several systems intended to empower children in spite of their deficiencies. We also examined systems that use storytelling as their main approach. This direction deserves additional research. We have explored the potential relationship between the development of children's narrative skills and six forms of programming activity. A system that could leverage the power of storytelling and advance a child's understanding of both programming and narrative simultaneously could be a powerful educational tool. We are currently developing one such tool, called Kokopelli's World, and describe it elsewhere (Thompson et al. 2016).

9. Acknowledgements

The authors would like to thank the reviewers for their helpful comments. This research was supported in part by grant P50HD071764 from the Eunice Kennedy Shriver National Institute of Child Health and Human Development (NICHD) at the National Institutes of Health (NIH). Thanks also go to Virginia Berninger for leading our center and encouraging the development of Kokopelli's World.

10. References

M. Andrade. (2007) Floresta Magica 2, product manual, Cnotinfor, Coimbra, Portugal. Referenced from the on-line version at http://arca.imagina.pt/manuais/Floresta%20Magica%202.pdf, retrieved on July 30th, 2016.

A. N. Applebee. (1978) *The Child's Concept of Story: Ages Two to Seventeen*. Chicago, IL: University of Chicago Press.

A. F. Blackwell. (2014) What is programming? *Psychology of Programming Interest Group*, 204-218.

M. Conway, R. Pausch, R. Gossweiler, and T. Burnette. (1994) Alice: A rapid prototyping system for building virtual environments. In *Conference Companion on Human Factors in Computing Systems* (CHI '94), Catherine Plaisant (Ed.). ACM, New York, NY, USA, 295-296.

S. Dasgupta, W. Hale, A. Monroy-Hernandez, B. M. Hill. (2016) Remixing as a pathway to computational thinking. In *Proceedings of CSCW '16*. San Francisco, CA, USA, 1438-1449.

C. Duncan, T. Bell, and S. Tanimoto. (2014) Should your 8-year-old learn coding? In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (WiPSCE 2014), 60-69.

L. P. Flannery, B. Silverman, E. R. Kazakoff, M. U. Bers, P. Bonta, and M. Resnick. (2013) Designing ScratchJr: Support for early childhood learning through computer programming. *IDC '13 Proceedings of the 12th International Conference on Interaction Design and Children*, 1-10.

L. Gould and W. Finzer (1984) Programming by Rehearsal. Xerox Corp. Palo Alto Res. Center Report No. SCL-84-1, May 1984.

A. K. Hansen, H. A. Dwyer, C. Hill, A. Iveland, T. Martinez, D. Harlow, and D. Franklin. (2015) Interactive design by children: A construct map for programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (IDC '15). ACM, New York, NY, USA, 267-270.

B. M. Hill, and A. Monroy-Hernández. (2013) The remixing dilemma: The trade-off between generativity and originality. *American Behavioral Scientist*. 57-5, 643-663.

K. Howland, and J. Good. (2015) Learning to communicate computationally with Flip: A bi-modal programming language for game creation. *Computers & Education*, 80, 224-240.

InfoWorld (1983). Software Reviews: Spinnaker. InfoWorld Vol. 6, No. 33, 56-58.

K. Kahn. (1996) ToonTalk^{TM} – An animated programming environment for children. *The Journal of Visual Languages and Computing*, April 16, 7(2), 1-23.

C. Kelleher, and R. Pausch. (2007) Using storytelling to motivate programming. *Communications of the ACM*, 50(7) p. 58-64.

LCSI, Logo Computer Systems Inc. (2004) MicroWorldsJR – MicroWorldsJR Resource Book with Extended Reference Guide, ISBN 2-89371-546-X, LCSI, Highgate Springs, Vermont, USA. Electronic book at http://www.lcsi.ca/pdf/microworldsjr/mwjr-resource-extended.pdf, retrieved on July 30th, 2016.

R. Lister. (2011) Concrete and other neo-Piagetian forms of reasoning in the novice programmer. *Thirteenth Australasian Computing Education Conf. (ACE 2011)*, Perth, Australia, 9–18.

O. Meerbaum-Salant, M. Armoni, and M. Ben-Ari. (2011) Habits of programming in Scratch. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education* (ITiCSE '11). ACM, New York, NY, USA, 168-172.

L. Morgado. (2005) Framework for computer programming in preschool and kindergarten. Ph.D. dissertation, University of Trás-os-Montes and Alto Douro, Portugal.

L. Morgado, M. Cruz, and K. Kahn. (2006) Radia Perlman – A pioneer of young children computer programming. *Current Developments in Technology-Assisted Education*, 1903-1908.

M. Pahud (2016) Insights from exploration of engaging technologies to teach reading and writing: Story Baker. In *Proceedings of the Conf. on the Impact of Pen and Touch Technologies in Education*, Providence RI.

S. Papert. (1980) *Mindstorms: Children, Computers, and Powerful Ideas.* Basic Books. New York.

S. Papert, and C. Solomon. (1972) Twenty things to do with a computer. MIT, AI Lab. *LOGO Memo* 3, July 1971. Also in *Educational Technology*, April 1972.

J. Piaget and B. Inhelder. (1969) *The Psychology of the Child*. Basic Books.

C. Rader, C. Brand, and C. Lewis. (1997) Degrees of comprehension: Children's understanding of a visual programming environment. *Proceedings of the 1997 ACM SIGCHI Conference on Human factors in computing systems*, 351-358.

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

H. Simon. (1996) *The Science of Design*, in *The Sciences of the Artificial, 3rd Ed*. Cambridge, MA: The MIT Press.

N. Smith, C. Sutcliffe, and L. Sandvik. (2014) Code club: Bringing programming to UK primary schools through Scratch. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (SIGCSE '14). ACM, New York, NY, USA, 517-522.

M. Stadler, and G. C. Ward. (2005) Supporting the narrative development of young children. *Early Childhood Education Journal*, 33:2, 73-80.

S. L. Tanimoto, and M. S. Runyan (1986) PLAY: An iconic programming system for children. In S.-K. Chang, T. Ichikawa, and P. A. Ligomenides (eds.) *Visual Languages*. New York: Plenum Press, 191-205.

R. Thompson, S. Tanimoto, V. Berninger, and W. Nagy. (2016) Coding, reading, and writing: Integrated instruction in written language. *Visual Languages and Human Centric Computing 2016*. (to appear).

L. S. Vygotsky. (1962) *Thought and Language*. Cambridge, Mass.: M.I.T. Press.

D. T. Willingham. (2006) Ask the cognitive scientist: The privileged status of story. *American Federation of Teachers*, http://www.aft.org/periodical/american-educator/summer-2004/ask-cognitive-scientist.

J. M. Wing. (2006) Computational thinking. *Communications of the ACM,* 49:3, 33-35.