How Programming Fits With Technology Education Curriculum

By Geoffrey A. Wright, Peter Rich, and Keith R. Leatham

There is an integral [technological] literacy component that needs to be more clearly identified, discussed, and included within STL*—computer programming.*

Programming is a fundamental component of modern society. Programming and its applications (e.g., Web and game development, online communications, networking, and storage) influence much of how people work and interact. Because of our reliance on programming in one or many of its applications, there is a need to teach students to be programming literate. Because the purpose of the International Technology and Engineering Educators Association (ITEEA) is technological literacy, it follows that technology education teachers include programming literacy as one of the fundamental literacy domains they teach. In this article, we advocate that programming literacy be taught in school and demonstrate how it fits within ITEEA's *Standards for Technological Literacy: Content for the Study of Technology* (*STL*) (ITEA/ITEEA, 2000/2002/2007) framework. In order to understand how teachers might incorporate programming literacy into their existing courses, we then provide a practical example of a current junior high teacher who has modified his communications courses to incorporate programming literacy through the design of videogames.

Programming Literacy is Part of Technology Education

Much of what is taught in technology education stems from ITEEA, which defines the purpose of technology education as technological literacy. In the seminal work, *Standards for Technological Literacy,* (*STL*), ITEEA outlines what are believed to be the "essential core of technological knowledge and skills we might wish all K-12 students to acquire" (pp. v, ITEA/ITEEA), stating that the content standards listed in *STL* are "what students should know and be able to do in order to be technologically literate" (pp. vii, ITEA/ITEEA). ITEEA wrote that the *STL* content standards resulted from the need to continue the development and evolution of technology education into technology and engineering education, stating,

> "Technology and engineering teaching has evolved as technology has advanced. During the industrial era of the 20th century, it was taught in the schools as indus-

trial arts, reflecting the industrial society. As advancements have catapulted us into a faster moving, more highly sophisticated technological society, technology and engineering education has made content adjustments that reflect these changes" (ITEEA, 2011, www.iteea.org).

ITEEA suggests that *STL* outlines the requisite content adjustments. Notwithstanding, ITEEA notes in *STL* that the content standards "will [and should] undergo periodic reassessment and reevaluation. It is very much a living document" (pp. vi, ITEA/ITEEA). In this paper we identify a critical omission to these content standards—computer programming—and argue for its more central inclusion in the standards.

Because the purpose and scope of technology education is technological literacy, and because the definition of technological literacy is subject to technological evolution and change, there is an integral literacy component that needs to be more clearly identified, discussed, and included within *STL*—computer programming. Curiously, the word "programming" is not referenced within the current *STL* document; however, there are a few *STL* standards that imply a need to teach programming literacy. For example, standards one through six suggest that students understand the scope, core concepts, and relationships a particular technology has with other fields, and its social, economic, political, and environmental impact on the world. Programming has a significant relationship with all of the fields of technology outlined in *STL* (for example, programming influences the fields of manufacturing, medicine, and communication by providing IT support and software development)—yet programming is not mentioned, leaving us to question how a particular technology is fully understood and appreciated without understanding one of its key relationships (programming). Although there are other domains that are not represented in *STL* (ex. CAD) this paper only focuses on programming—as the authors believe it is one of the most obvious omissions and fundamental to technological literacy.

An understanding of how programming influences the world will help students appreciate the significant role and influence programming (and its various applications) has on economics, politics, the environment, and the social aspects of the world.

STL standards eight through eleven also have some pertinence to programming literacy. These standards discuss the need for students to understand and apply the attributes of the design process, the role of troubleshooting, research and development, invention and innovation, and experimentation in problem solving. Again, each of these topics is important to the field of programming, especially problem solving (Steffe, Nesher, Cobb, Goldin, & Greer, 1996; de Corte, Verschaffel, & Schrooten, 1992), yet programming is neither mentioned nor advocated in the document.

Finally, *STL* Standard 17 mentions a few terms that imply the need to teach programming and have students be programming literate. For example, "language is key to communication" (ITEA/ITEEA, pp. 167), and "computers are at the center of the Information Age and are the primary tools used" (pp. 167). These references, however, only have a vaguely implied orientation to programming—which is insufficient considering the world's increasing dependence on programming. The focus of *STL* Standard 17 is on the broad domains of information and communication technologies. Although this focus is helpful and pertinent to a general understanding of technology, what is listed in *STL* ("Students will develop an understanding of and be able to select and use information and communication technologies" (ITEA/ITEEA, pp. 166) by no means clearly identifies or highlights the demand and need to be programming literate. Because of our dependence on digital communication, there is a significant need to be programming literate. The next section presents an argument to support this claim.

Why Programming is Important

The Computer Science Teachers Association (CSTA) offers four reasons that computer science is important in today's society. Namely, that computer science links to other fields, it teaches problem solving, it can engage all students, and it leads to greater employability. In the following section, we replace "computer science" with programming literacy. Our goal is no more to make every student into a computer scientist than a middle-school social studies teacher's goal is to turn every student into an historian. We recognize and support student diversity and interest across the curriculum. Yet, we acknowledge the changing world in which we live and the need to better equip students to compete confidently in an increasingly technological society. Thus, our goal is for every student to become programming literate. In the following section, we explain what we mean by programming literacy and why we feel it is an important understanding for students to gain.

Programming Literacy

Literacy has always been a core tenet of a successful society. Throughout history, the ability to communicate and achieve societal goals through reading and writing has been a skill that has provided career and lifestyle security. Dating back to early American settlements, the ability to read and write provided the literate an advantage over those who were less literate. This same structure continues in today's business world.

According to the Organization for Economic Co-operation and Development (OECD), the international organization tasked with assessing and comparing students' academic abilities in over 50 different countries, "…reading literacy is understanding, using, and reflecting on written texts in order to achieve one's goals, to develop one's knowledge and potential, and to participate in society" (2003, p. 108). Thus, a literate individual is not one who can simply use a particular object, but rather someone who can manipulate the object in a meaningful way. Just because someone can read doesn't make him or her literate. Prensky (2008), author of *Teaching Digital Natives—Partnering for Real Learning*, wrote about this programming phenomenon stating,

> "I believe the single skill that will, above all others, distinguish a literate person is programming literacy, the ability to make digital technology do whatever, within the possible, one wants it to do—to bend digital technology to one's needs, purposes, and will, just as in the present we bend words and images. Some call this skill human-machine interaction; some call it procedural literacy. Others just call it programming" (¶ 4).

In short, Prenske argued that programming is a new type of literacy. Indeed, we might argue that the traditional 3Rs—Reading, Writing, Arithmetic—should be expanded to include "Programming."

Programming literacy is defined as being able to effectively, efficiently, and safely interact, use, and manipulate communication technologies (Prensky, 2008; Miller, 2004; Van Rossum, 1999). Technologies within this domain consist of: telecommunication devices (phones, phone applications, VOIP technologies), digital viewing and listening devices (mp3 devices, computers, computer software, gaming consoles), and web technologies (websites, chat rooms, online video games, search engines and browsers, online software, e-commerce, e-banking and investing, e-learning, etc.). Within each of these communication technology domains, a programming-literate individual will be empowered with the ability to more effectively and efficiently use the technologies because he or she understands how to manipulate them. Programming literacy can contribute to an individual's well-being in several ways, such as increased job opportunity, increased awareness of security issues, and increased ability to interact with and successfully manipulate one's environment.

Increased Opportunity

Being programming literate opens doors. According to the U.S. Department of Labor, "almost all of these careers [fastest-growing occupations] are in computers or healthcare" (www.dol.gov). On the Department of Labor website, seven of the top ten highest-demand occupations are computer-related positions. Additionally, it's rare to find an occupation, company, or job of any level that doesn't require or rely on programming at some level (e.g., networks define and control how companies work, most companies have IT or CSR support staff, most companies use software for financial reasons, to perform tasks, and so forth).

Jon Katz (2000) demonstrated how "geeks" (i.e., those with computer programming skills) "are literally building the new world economy" (p. xxiv). The programming-literate are those who are in positions of power in today's society. Katz, former executive producer of the *CBS Morning News*, told a story of being a guest on another anchor's news show. In response to a comment he made about the Internet, two cameramen and several technical controllers began clapping and cheering—on live television! Off-camera, Katz asked his friend how they could get away with something like that without getting fired. His friend, clearly flustered, replied, "How could we fire those guys? Nobody else can run the… place!" (p. xxi).

Increased Ability to Successfully Manipulate One's Environment

Programming, in its most basic form, involves the designing, scheduling, and planning of an event or events. Therefore a programming-literate individual would be someone who is able to digitally customize his/her life by designing, scheduling, and planning events using programming. Examples of this would be someone who programs digital devices to facilitate or enhance daily tasks, workloads, research projects, production, and so forth.

Key to this discussion is the ability to manipulate. Manipulation of communication and information devices centers on an understanding of programming. *STL* Standard 17

discusses the need for a technologically literate individual being able to understand that "communication systems are made up of a source, encoder, transmitter, receiver, decoder, and destination" (ITEA/ITEEA, 2000/2002/2007). However, just understanding that communication systems are made up of these pieces does not mean an individual can manipulate these things. Manipulation of an encoder would mean a student understands the language of how the digital files are encoded, and the subsequent ability to change or modify that encoding, "to achieve one's goals…and to participate in society."

People increasingly want to have more control and function of certain devices, and want other devices to be more automated. Consider homeowners who now have programmed lighting in their homes that comes on at specified times, people who have their email or RSS readers automatically aggregating news feeds, students and adults who customize their Facebook accounts to automatically update calendars, birthday invitations, news feeds, personal updates, scores and statistics for work and or leisure, and so forth.

Just as in years past where reading and writing literacy provided people lifestyle security, today there is a need to ensure that people are programming literate, that they can read and interact with a world created and manipulated by programmers. Not only would this protect them from online scams, but it would also provide them with marketable skills, enabling a more secure online lifestyle and career. Programming should be taught because it plays an integral part in everyday life; its influence inundates the business world, academics, politics, commerce, banking, even leisure activities. Because technology educators have the need to teach and promote technological literacy, it is imperative that they teach the one aspect of technology that influences and has a relationship with all of the activities of everyday life.

How Teachers Can Incorporate Programming Literacy as Part of Technology Education Curriculum

Just as the *STL* standards do not prescribe or "define a curriculum, for the study of technology" stating that this "is something best left to states and provinces, school districts, and teachers" (ITEA/ITEEA, 2000/2002/2007), this paper will not dictate what exact procedures or tools should be taught. Rather, its intent is to highlight the need to include programming literacy in technology education, and to suggest that if the intent of technology education (or technology and engineering education) is to promote technological literacy, programming literacy needs to be included as one of the central abilities for a technological world (much like construction technology, manufacturing technology, medical technology, and so forth are included).

The last part of this article highlights the experience of a teacher who has started a grassroots movement to incorporate programming literacy into his technology and engineering education curriculum at the junior high school level. The intent of highlighting this teacher is to provide an example of what can be done to integrate programming literacy into an already existing technology class where the technological literacy standards are the basis of the curriculum, and to discuss his process, methods, rationale, and the benefits of doing so.

Introducing Programming Literacy Into a Communications Class

Ryan Anderson is a junior high technology teacher in Alpine, UT. He graduated from a Technology and Engineering Education teacher preparation program. Although his preservice training did not include much programming, he understood the need for students to be programming-literate and was equally aware of their interest in that domain. Consequently, he teamed with a group of educators from his former university to include a unit on programming literacy in his curriculum. The curriculum was game-design-based, where students would use a programming tool (in this case CS5 Flash) to build simple Internet games. Research has demonstrated that the design and development of games is an extremely effective technique for helping students to learn (Torres, 2009). Even the simplest of games (e.g., tic-tac-toe) may require higher-order thinking skills and allow students to practice applying content and skills learned in other classes (e.g., algebra). Ryan structured the curriculum using a scaffolded design approach, wherein portions of the code were completed (and heavily commented) for students who completed the game by applying increasingly advanced programming skills. Table 1 demonstrates the curriculum map used. The following is a summary description of one of the lessons.

Day 4: Lesson 7 – Variables

The class starts with Ryan providing an anticipatory set introducing variables. The anticipatory set involves having an empty box, labeled "variable," on a table in the front of the classroom. Ryan explains to the students that the box could hold any assortment of things and proceeds to put a base-

Table 1
Scaffolded Programming Curriculum and Weekly Schedule

LESSON	ASSIGNMENTS
Days 1-3 Lesson Zero – Flash GUI 1. Students will understand the timeline. 2. Students will understand layers. 3. Students will be able to use the drawing tools. 4. Students will create a splash screen.	Splash Screen for Duck Hunt
Day 4 Lesson One – Intro to ActionScript 1. Students will understand why they would want to learn ActionScript 3.0. 2. Students will manipulate the order of their movie using timeline controls.	Duck Hunt Symbols
Day 5 Lesson Two – Symbols 1. Students will understand object types and instances. 2. Students will know how to communicate with instances from ActionScript. 3. Students will be able to modify properties of instances on the stage. 4. Students will understand the coordinate system of the stage. 5. Students will know how to use the Adobe Help.	Duck Hunt Symbols
Day 6 Lesson Three – Interactivity 1. Students will be able to define function. 2. Students will be able to write their own functions. 3. Students will understand EventListeners. 4. Students will know how to add an EventListener to any object on the stage. 5. Students will understand the importance of commenting their code.	WEB PAGE or STORY BOOK
Day 7 Lesson Four – Variables 1. Students will be able to define "variable." 2. Students will be able to declare their own variables. 3. Students will understand variable types. 4. Students will know how to export an object from the library and add an instance to the stage. 5. Students will be able to use trace statements to debug their code.	ZELDA DODGE GAME
Day 8 Lesson Five – Strings and Text Boxes 1. Students will be able to define "string." 2. Students will understand the difference between static and dynamic text boxes. 3. Students will be able to change the value of a string using functions.	LOUCO LIBS
Day 9 Review and Catch-Up Day	
Day 10 Lesson Six – Using Math 1. Students will understand order of operations. 2. Students will be able to use Math.random(); 3. Students will know all of the main arithmetic operators of ActionScript.	Worksheet
Day 11 Lesson Seven – Boolean Logic 1. Students will understand if/else statements. 2. Students will be able to write their own if/else statements. 3. Students will explore uses of Boolean variables. 4. Students will know how to use Boolean operators.	Worksheet
Day 12 Lesson Eight – Calling Objects 1. Students will understand event.target and event.currentTarget. 2. Students will understand what a For Loop is used for. 3. Students will be able to write their own For Loops.	DRAG & DROP GAME
Day 13 Lesson Nine – For Loops Conceptual Objectives: 1. Students will understand why loops are used in programming. 2. Students will be able write their own For Loops.	FISH GAME
Day 14 Lesson Ten – Special Events 1. Students will be able to use TimerEvents. 2. Students will understand the ENTER_FRAME event.	DUCK HUNT GAME
Days 15-20 Lesson Eleven – FINAL GAME	A GAME OF THEIR CHOICE (Suggestions given in Lesson Plan)
Day 21 PARTY/SHOWCASE of Games and Code-Walk	Students showcase their games for peers, parents, teachers, and administrators. The intent is for students to talk about the programming, explain the code, and finally discuss the impact programming has on society.

ball glove in the box. He tells the students this variable (the box) has a baseball glove in it. He then takes out the baseball glove, puts in a can of soup, and says, "The variable now has a can of soup in it." He tells the students that variables in programming, and even math, work the same way. Variables can be any object or number. At this point he makes an explicit connection to math, stating that a variable in programming, like math, simply refers to a symbolic name of an object that has some value. Ryan then proceeds to write several mathematical equations and programming expressions on the board. Two examples he writes are:

Example 1: 2x + 3y = 10; f(x) = 4, solve for y.
Example 2: var numberOne:Number = 5;
trace(numberOne)

Ryan then proceeds to reiterate what variables are, asking students to define variables for each other. The second part of the lesson involves the students opening up the Flash software to complete a variable assignment with a partner. One partner is to "drive the computer" (control the mouse) while the other student is to tell him/her what to do. The "driver" is not allowed to complete any task without being told what to do. The assignment requires the students to create three variables that will hold an instance of a particular object. They are then asked to program the objects to do three different things based on the variables they created. One object is to move to the top right of screen, the second object is to increase in size, and the third object is to disappear. After successfully completing the assignment, the students are then provided an independent assignment where each student is to take a partially completed game that focused on writing and using variables and complete the game by creating and inserting variables in the appropriate locations. When students complete this assignment, they will have created a simple working game that uses variables. At the end of the class, Ryan asks a few students to show their code and talk about the variables they created—and what the variables do. For homework, the students are asked to continue programming their individual games and adding to them as they learn new concepts. In this case, they are now expected to add variables to their game.

A key component of this lesson is how Ryan builds upon prior knowledge and scaffolds in various previously learned ideas from programming AND math. For example, he explicitly discusses the topic of variables and how they pertain to mathematics and programming. He also ensures that he is building upon prior knowledge in a logical manner, where each topic supports and hinges upon understanding a previously learned concept. For example, prior to teaching variables, he ensures students are familiar with basic programming structure, functions, and best practices of writing code.

Ryan has discovered several benefits from implementing programming into his technology curriculum. First, he has learned the basics of programming. He feels he has become programming-literate and believes that, because programming is entwined everyday life, he has a responsibility to understand it—and teach it. Ryan learned the basics of programming by collaborating with a student teacher who was assigned to him and by reading various tutorials.

A second benefit Ryan has noticed since implementing the programming unit in his curriculum is how excited students are about the topic (he has reported that his class has become the most popular elective course at the school, which he says, "is due in large part to teaching kids how to make video games in Flash") and how be believes the students have become more critical thinkers. Ryan has noted that the students are now able to go to the Internet to find coding and quickly interpret it and implement it in their own games. He says the kids have taken the basics that he has taught them and quickly surpassed his own abilities, making fully functional multiplayer games with hundreds of variables and functions. Ryan believes this type of excitement and critical learning will help the students in all of their other studies.

Conclusion

Programming literacy is essential in today's heavily computer-dependent society. Being programming literate may increase critical-thinking and problem-solving abilities—and as Ryan noticed in his classroom, teaches a core concept of how things work in today's society and may help promote technological developments. Consider Papert's (1993) observation:

> "Although the software that can be purchased today gives only an inkling of what is to come, it should be seen in the same light as the first flight of the Wright Brothers' machine. Its importance for the future was not measured by its performance in feet of flight, but its ability to fuel the well-informed imagination" (¶ 17).

As society moves into a new era further dominated by computers, being programming-literate will ensure job security, personal security, and help advance technological developments. Sadly, programming is not being pushed in K-12 curriculum structures. Schollmeyer (1996) observed, "Although

there exists a computer science certification exam for high school teachers in some states, there appear to be no general requirements for teaching computer science at the high school level in most states" (pp. 378). Thus Papert's (1993) observation concerning the need to teach essential technologies in schools still applies today in 2011: "The most important consequences of new technologies are not recognized by education policy-makers" (¶ 1).

Despite the lack of programming inclusion in mainstream K-12 curriculum, there are various entities that, like Ryan, are trying to make movements to promote programming among K-12 students. For example Google© has launched several "code contests" and camps where students can learn and compete in various programming-related courses and competitions. Google has said these courses are important because, "With a world that is increasingly reliant on computer technology, encouraging students to explore programming is crucial" (Waters, 2010, ¶ 4).

Programming literacy is important, and technology educators should join with computer science teachers in introducing it in schools. The technological literacy standards suggest that technology educators should teach the technologies that will have the greatest impact on society, politics, economics, and environment: programming is one of these technologies.

References

De Corte, E. (1992). On the learning and teaching of problem-solving skills in mathematics and LOGO programming. *Applied Psychology. An International Review, 41*(4), 317-331.

ITEA/ITEEA (2000/2002/2007). *Standards for technological literacy: Content for the study of technology.* Reston, VA: Author.

Katz, J. (2000). *Geeks*. New York: Broadway Books.

Miller, J. A. (2004). *Promoting computer literacy through programming Python.* A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Education) at The University of Michigan.

OECD. (2003). *The PISA 2003 assessment framework—mathematics, reading, science and problem-solving knowledge and skills.* Organisation for Economic Co-operation and Development.

Papert, S. (1993). Obsolete skill set: The 3 Rs—Literacy and letteracy in the Media Ages. *Wired* magazine. Retrieved from: www.papert.org/articles/ObsoleteSkillSet.html

Prensky, M. (2008). Programming is the new literacy. Published in the Feb. 2008: Global Education issue of *Edutopia* magazine as "Programming: The New Literacy." Retrieved from www.edutopia.org/programming-the-new-literacy

Schollmeyer, M. (1996). *Computer programming in high school vs. college.* Proceedings of the 27th SIGCSE technical symposium on Computer Science Education (pp. 378-382). New York. ISBN: 0-89791-757-X.

Steffe, Leslie, P. et al, Eds. (1996). *Theories of mathematical learning.* Mahwah, NJ: Lawrence Earlbaum Associates.

St. George, D. (2010). *Race to nowhere (*film highlights stress students face in high-pressure academics). Retrieved from www.washingtonpost.com/wp-dyn/content/article/2010/10/07/AR2010100702713.html

Torres, R. J. (2009). *Learning on a 21st century platform: Gamestar mechanic as a means to game design and systems-thinking skills within a nodal ecology.* Thesis, New York University: ProQuest Dissertations.

van Rossum, Guido. (1999). *Computer programming for everybody: A scouting expedition for the programmers of tomorrow.* Corporation for National Research Initiatives.

Watters, A. (2010). *Google launches contest to encourage kids to code.* Retrieved from www.readwriteweb.com/archives/google_launches_contest_to_encourage_kids_to_code.php

Geoffrey A. Wright, Ph.D. *is an assistant professor of Technology Engineering Education in the School of Technology of the College of Technology and Engineering at Brigham Young University in Provo, Utah. He can be reached via email at ge.wright@gmail.com.*

Peter Rich, Ph.D. *is an assistant professor of Instructional Psychology and Technology in the School of Education at Brigham Young University. Dr. Rich's Ph.D. is in Educational Psychology and Instructional Technology from the University of Georgia, Athens.*

Keith Leatham, Ph.D. *is an associate professor of Mathematics Education in the College of Computing Science and Mathematics at Brigham Young University. Dr. Leatham received his Ph.D. from the University of Georgia, Athens in Mathematics Education.*

This is a refereed article.