PROGRAMMING FOR PRE-COLLEGE EDUCATION USING SQUEAK SMALLTALK

Kathryn N. Rodhouse
Missouri University of Science & Technology

Benjamin Cooper
Savant, LLC

Steve E. Watkins
Missouri University of Science & Technology

Abstract

Competence in a programming language can provide a strong basis for logical thinking and an exposure to technology; however, many languages are perceived to be too complicated to learn at a young age. Opportunities for pre-college students to learn programming concepts can help develop critical thinking and problem solving skills that will enhance their educational experiences. Also, conceptual understanding of programming techniques in one language can aid in learning other languages. This project developed an integrated series of programming tutorials for using Squeak Smalltalk. Squeak Smalltalk is an open-sourced, object-oriented programming language that is being used for educational software and through the One-Laptop-per-Child initiative as well as for database and multi-media applications. The intent of the tutorials is to allow anyone, no matter their current programming proficiency, to learn applied programming techniques and to avoid simple manipulation of code without understanding the underlying concepts. These tutorials cover object-oriented commands, conditional coding, programming methods, variables, and classes in the context of Squeak Smalltalk, but these concepts are applicable to many languages. The object-oriented nature of Squeak Smalltalk facilitated the development of basic programming literacy. The tutorials, an implementation with pre-college students and teachers, and an associated assessment are described. Completion of the tutorial series enabled the young programmers to adapt the Squeak software package for their own original programs.

Introduction

Programming literacy is an important component in educating a modern workforce and has particular relevance for those pursuing science, technology, engineering and mathematics (STEM) careers. Programming promotes the development of logical thinking and problem solving, both of which are skills necessary for success is many technical fields. The National Science Foundation projected that there would be a shortfall of natural science and engineering bachelor degrees in the year 2006.[1] The United States needs to remain technologically advanced in order to compete in world markets.[2] In April of 2004, the U.S. Education Department's National Center for Education Statistics reported that fewer than half of seniors in high school were taking a science course, which emphasizes the fact that there is a decline of interests in scientific fields within the United States.[3] Students need to be exposed to technical topics at earlier ages before they decide that STEM-related subjects are uninteresting or too hard. Many educational programs are currently seeking new methods to improve STEM curriculum.[4] Additionally, many new extracurricular programs such as Best Robotics are developing with the sole purpose of attracting students to technical fields.[5] Programming literacy efforts are included in these efforts.

In most college engineering programs, an introductory software programming course is required. Programming skills and an understanding of software are important in many engineering professions.[6] Programming tasks require that a problem be explicitly defined, that a possible solution be formulated, that the solution be implemented, and that the solution be tested. This problem-solving process is relevant to most engineering work. Early programming skill development enables students to develop these critical skills, and it helps them gain computer experience that can directly apply to a technical field.[7] By exposing students to programming in an interesting way, students may be encouraged to join programs and competitions such as those hosted by the American Computer Science League,[8] which are also aimed to further interest in mathematics, science, and technical fields.

This project demonstrates the development of programming tutorials based on Squeak Smalltalk for pre-college audiences. These tutorials cover object-oriented commands, conditional coding, programming methods, variables, and classes in the context of Squeak Smalltalk, but these concepts are applicable to many languages. The intent of the tutorials is to allow students with little or no programming experience to learn and apply programming techniques. The approach exploits the graphical nature of Squeak Smalltalk and avoids simple manipulation of code without understanding the underlying concepts. The need for pre-college programming experience is described as well as the Squeak Smalltalk environment, an overview of the tutorials is given, and a preliminary assessment with middle-school-age and high-school-age students is given. The results indicate that these young students are capable of learning and enjoying programming.

Programming Literacy

Traditionally collegiate engineering courses are taught using FORTRAN, Basic, or C++; however, recently some collegiate programs have been experimenting with different type of development languages. One such program at the Citadel has been implementing Mathcad to teach students the basics of programming.[7] This experimentation was prompted by the search for a programming language that could enable students to learn the basics of programming syntax while also learning how to use programming to solve a problem. Oftentimes engineering courses are largely focused on language specific syntax that does not enable cross language feasibility and does not teach the process to solving a programming problem.

In addition to experimenting with languages, many colleges are searching for better methods of teaching students to program. Because many engineering students lack experience with programming, oftentimes these students also lack the ability to finish programming assignments in a timely manner. Finding ways to help students outside of the classroom have been a recent goal for many engineering programs. One method used by the University of Cincinnati requires that a Programming Learning Center be implemented.[9] This Learning Center allowed students to observe seminars about the assignments and to ask questions as needed. This particular Learning Center allowed 50% of the programming students to complete their assignments on time whereas without the Learning Center only 2.4% of students completed the programs on time. This statistic shows that any additional guidance can largely benefit students in understanding the processes of programming.

At Pennsylvania State University, educators have found that interactive and hands-on lecturing styles are also aiding students in the classroom.[10] By creating lecture notes that can be viewed and adapted individually during and after the class, students were capable of viewing and working through the notes at their own speed, allowing them to learn more from lectures.

Because there is such a strong need for the development of good programming skills in engineers, educational systems are constantly looking for methods of course improvement. By learning from past methods, courses are being changed and adapted to better aid students in learning the process of programming.

Squeak Smalltalk Environment

Squeak Smalltalk is the language used for this project. Squeak Smalltalk is an open-sourced language that is currently being utilized for many applications including multimedia applications targeted to the pre-college classroom.[11] This programming environment is based on the original Smalltalk programming language that was developed as Xerox PARC in the 1970s. The environment has been used for the One-Laptop-per-Child initiative as well as other education and recreational applications, but it is fully function with other commercial applications such as database and multimedia tools. A key characteristic for the pre-college audience in this project is the object-oriented nature of Squeak Smalltalk.

The intent of the project is to provide a first-experience in programming for pre-college students. Additionally, the experience should provide an organized approach to programming and develop an understanding of key programming concept. The Squeak Smalltalk environment is well suited for younger audiences due to its visual nature. It can illustrate the processes of programming without having to concentrate heavily on syntax understanding. It is an excellent vehicle for learning the basic terminology and theory of programming.

Squeak Smalltalk Tutorials

A series of tutorials were developed to familiarize users with the Squeak Smalltalk interface and basic programming syntax. This series consists of five tutorials. Each tutorial allows users to create a visible change to their user interface. These tutorials were written in a manner to allow anyone, no matter their current programming proficiency or experience, to learn applied programming techniques. Completion of these tutorials will allow the programmer to adapt the Squeak classes and objects to do what they intend them to do. Furthermore, the tutorials introduce an organized approach to programming and key programming concepts that are used in any programming environment. Manipulation of code without understanding the underlying concepts is discouraged. In particular, the tutorials are intended for audiences as young as middle school and high school students. The visible, hands-on aspects enhance the Squeak programming experience for young users, who are able to see the objects they are programming.

Each tutorial contains basic information to complete selected visible tasks. The tutorials are designed to allow students to learn independently. During the preliminary implementations, a teacher was present to answer questions, but no formal presentation was made of the material. Additionally, the tutorials give background information about the theory of programming and how typical programming languages work. Sidebars and "Did You Know" Sections are included with an assortment of shortcuts and interesting facts to make them both more informative and more interesting. The summary page from the second tutorial is shown in Figure 1. Note the sidebar on getting more information on Squeak. Special consideration is needed for the intended young audience.

- The guided tasks must be divided into segments with a single concept.
- The tasks allow students to do something with each concept before continuing.
- The terminology must be clearly defined and consistent throughout the tutorials.
- Students are encouraged to try variations while working at their own pace.
- Connecting concepts and key information are repeated.
- Sidebar and "Did You Know" information highlights selected facts, resources, etc.

Tutorial 2: Summary.

In this Tutorial, we learned how to use the Repeat conditional and the Random object to make designs with the Pen tool we learned to use in Tutorial 1.

We can use the skills we learned in this Tutorial to draw different figures and polygons. We can use the commands we learned to draw pictures similar to how we could draw pictures with a spinning pen.

Keep testing this out and use the skills you learned in this lesson to draw different figures in your Squeak project.

In the next Tutorial, we will learn how to make our own commands for objects in Squeak. We will learn how to make some directional commands for the Pen tool.

We will even use some of our favorite designs that we made in this tutorial to turn into a command. By doing this, each time we call the command through a Pen object it will make our favorite designs in our project.

Tidbits

For more information about Squeak, visit squeak.org.

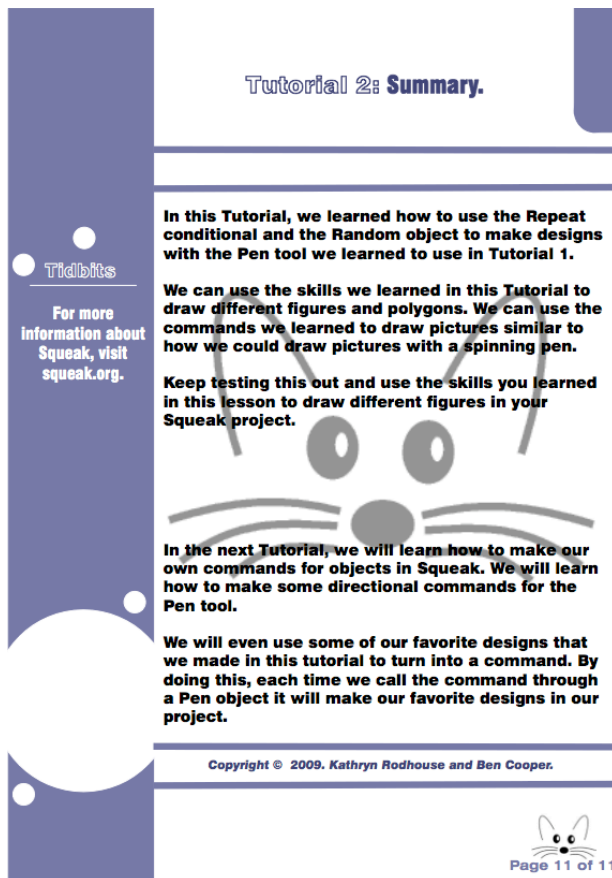Copyright © 2009. Kathryn Rodhouse and Ben Cooper.

Page 11 of 11

Figure 1: Example Summary
Page from Tutorial 2.

The logo and the color scheme are designed to appeal to the young audience. The organization of the five tutorials with the selected tasks and the associated programming concepts are shown in Table 1.

The purpose of the initial tutorial is to familiarize students with the built-in structure of Squeak Smalltalk and to spark an interest for completion of future tutorials. These tutorials guide students to create a project and to use objects built into the Squeak software to manipulate the user interface. By the end of this tutorial, students create a drawing tool with their program similar to an "Etch A Sketch". An example of the task results is shown in Figure 2. Both reference information about Squeak Smalltalk and basic programming rules are included.

The second tutorial teaches students about conditional programming statements. Students draw on their user interface using Squeak commands and repeat the commands with conditional statements, which then create a "spinning pen" type tool. In addition to learning

Table 1: Organization of the Tutorials.

Tutorial	Tutorial Tasks	Concepts Taught
1. Getting Started in Squeak	Use Built-in Squeak Objects Create an Etch-A-Sketch" type Tool	Basic Programming Rules Basic Squeak Historical Information
2. Introduction to Conditionals and the Random Function	Create a "Spinning Pan" Tool	Conditional Programming Statements Computer Graphic Display Information History of RPG Displays
3. Making Commands in Squeak	Develop and Store Original Programs Manipulate Squeak Versions	Embedded Code Absolute Commanding Relative Commanding
4. Making Advanced Commands in Squeak	Develop Complex Programs and Commands Use a Pen to Draw Intricate Designs	Embedded Code Conditional Statements Debugging Concepts
5. Making Objects in Squeak	Develop and Store a Class	Inheritance More Squeak Historical Information

Figure 2: Screen from Drawing Task (Student have Choices of Color, Shape, etc).

about conditionals, students gain background information on how computers display colors through completion of this tutorial. In the third tutorial, students learn how to develop their own programs and store them into the Squeak software. Students are able to manipulate their version of Squeak to store their own built-in commands. Through this tutorial, students learn the difference between absolute and relative commanding and the meaning of embedded code. The fourth tutorial develops student skills with more complex commands and saved programs. Students explore conditionals and embedded code further in this tutorial. Visually, students make programs to command a "Pen" to "draw" designs (a star design is demonstrated within the tutorial). Debugging concepts are also taught in this tutorial. The final tutorial in this series allows students to program and define

a class. Students learn the concept of inheritance and how to utilize it in order to "reuse" built in programs and functions of the Squeak software. More historical information concerning Smalltalk is given within this tutorial.

This series of five tutorials cover basic programming concepts. After modification that is based on the assessments, a second series of tutorials are planned, i.e. Tutorials 6-10, that will provide more complex instructions with a focus on further programming process and theory development. In particular, an individualized screen saver task using the Squeak Smalltalk environment will allow students to explore proper conditional handling and proper Smalltalk syntax in addition to using skills learned from the original series of tutorials.

Implementation and Preliminary Assessment

A two-page tutorial evaluation was developed to assess the effectiveness of the tutorial design and to assess the usefulness of the tutorials for a pre-college audience. The background of the users is addressed in the initial section as shown in Figure 3. The effectiveness of selected aspects of the tutorial is addressed in the next section as shown in Figure 4. General assessment with Disagree/Agree items is provided as shown in Figure 5. The final section allows for open-ended responses to the following questions:

- How could these tutorials be improved to make them more interesting?
- How do you see yourself using the skills and ideas from these tutorials?
- Do you have any other suggestions for these tutorials?

A preliminary implementation of the tutorials was done at the Dent-Phelps R3 Public School, a rural K-8 school district in Missouri. The gifted class for fifth grade through seventh grade was given the tutorials in the school computer laboratory during a single two-hour period. Six students participated with ages from 10 years to 13 years old including four boys and two girls. All had received prior computer instruction, e.g. E-mail and word processing, through the school and most had computers available at home for E-mail and gaming. Only one had done limited programming before and none had used Squeak Smalltalk before. All completed Tutorial 1 and some started Tutorial 2 during the period. All participants felt that the first tutorial was beneficial and most (four) felt that it was "somewhat beneficial and give [them] a partial understanding of programming steps." Five out of six participants liked the tutorial sidebars. Most participants felt that the tutorial figures were helpful, but two participants did not like the current figures. The rating assessment results are summarized in Table 2.

SQUEAK SMALLTALK TUTORIAL EVALUATION PART 0

How old are you?   *Highest grade completed?*   *What is your gender?*
_____   _____   Male   Female   (circle answer)

Have you ever used Squeak before? Do you have a computer at home?
Yes   No (circle answer)   Yes   No (circle answer)

How proficient were you at programming, prior to using these tutorials? (Pick one.)
_____ very proficient, have programmed extensively in Squeak or other environments.
_____ somewhat proficient, have programmed before in Squeak or other environments.
_____ somewhat not proficient, have been introduced to programming topics.
_____ not proficient, have never been introduced to programming topics.

What is your general computer experience? (Select all that apply.)
_____ Browsing the internet and sending E-mail.
_____ Using word processing programs such as MS Word.
_____ Using spreadsheet programs such as MS Excel.
_____ Using graphics/drawing programs or photo processing programs.
_____ Playing computer games.

Which Squeak tutorials have you completed? (Circle all that apply.)   1   2   3   4   5

Figure 3: Background Section of the Tutorial Evaluation.

SQUEAK SMALLTALK TUTORIAL EVALUATION PART 1

Pick the best choice for each statement.
The tutorials were …
_____ beneficial and greatly helped me in understanding programming concepts.
_____ somewhat beneficial and gave me a partial understanding of programming steps.
_____ somewhat beneficial, but did not give enough detail.
_____ not helpful in understanding programming concepts.
The tutorial sidebars ("Did you know," "Tidbits," "How to,") ...
_____ had good information and advice that I read often.
_____ had some interesting information, but I did not always read them.
_____ provided some good tips for the tutorials.
_____ were uninteresting and did not benefit my tutorial experience.
_____ were uninteresting and I never read them.
The tutorial figures (showing computer screen examples for the Squeak projects) ...
_____ were helpful in understanding the tutorial steps.
_____ were somewhat helpful, but the tutorials had too many figures.
_____ were somewhat helpful, but the tutorials had too few figures.
_____ were not helpful in understanding the tutorial steps.
If you have used other programming languages, does Squeak programming seem …
_____ easier   _____ harder   _____ same difficulty   _____ not applicable

Figure 4: Second Section (Selected Items) of the Tutorial Evaluation.

SQUEAK SMALLTALK TUTORIAL EVALUATION PART 2

Please use the following scale to respond to each of the statements in Part 2:
Strongly Disagree 1 ... 2 ... 3 ... 4 ... 5 ... 6 ... 7 ... 8 ... 9 ... 10 **Strongly Agree**

_____ 1.   The figures were helpful by showing what the screen should look like.
_____ 2.   The length of the tutorials was sufficient to understand the concepts.
_____ 3.   The instructions for typing the code were easy to understand.
_____ 4.   I had to read the tutorials several times to understand the steps.

_____ 5.   I tried changing the given code to see what would happen.
_____ 6.   I had difficulty completing the tutorials.
_____ 7.   I gained a better understanding of programming terminology.
_____ 8.   The tutorial layout was well designed.

_____ 9.   I gained a better understanding of programming from these tutorials.
_____ 10. I feel comfortable creating my own code in the Squeak environment.
_____ 11. I want to continue my programming studies in the Squeak environment.
_____ 12. I would recommend these tutorials for others to learn about programming.

Figure 5: Third Section (Ratings) of the Tutorial Evaluation.

Table 2: Average Ratings from the Tutorial Evaluation.

Part 2 Question (Disagree 1 … 5 … 10 Agree)	Dent-Phelps R3 School	Home School Group	Park Hill South School
1. The figures were helpful …	7.50	8.50	8.35
2. The length … was sufficient to understand …	5.67	9.25	7.82
3. The instructions … easy to understand	5.50	8.50	8.29
4. I had to read … several times …	5.00	2.25	4.41
5. I tried changing the code (experimenting) …	8.67	8.00	6.59
6. I had difficulty completing the tutorials	4.50	1.50	2.59
7. I gained a better understanding … terminology …	6.17	8.50	6.06
8. The tutorial layout was well designed	6.83	8.25	8.00
9. I gained a better understanding of programming…	6.00	7.75	6.29
10. I feel comfortable creating my own code …	3.50	5.75	6.00
11. I want to continue (with) Squeak …	5.33	6.50	5.76
12. I would recommend these tutorials …	4.50	8.75	6.94

A second implementation of the tutorials was done for a home school group of students in California ranging in age 14 years to 18 years old. These participants completed all five tutorials during weekly two-hour sessions over five weeks. This group consisted of four students of which three were boys and one was a girl. All were frequent users of home computers and all had done some programming before. The group liked the tutorials with half selecting "beneficial" and half selecting "somewhat beneficial." They generally liked the sidebars and they all liked the figures. Those that had used other programming languages preferred Squeak. The rating assessment results are summarized in Table 2.

A third implementation of the tutorials was done at the Park Hill South School, a public school in Riverside, Missouri. The participants were sophomores, juniors, and seniors in a programming class (Java and Visual Basic) and they ranged in age from 15 years to 19 years old. The students completed at least the first two tutorials during a single 100-minute period. The group consisted of 19 boys. The group liked the tutorials with ten selecting "beneficial" and seven selecting "somewhat beneficial." They generally liked the sidebar information, but most (nine) "did not always read them."

Thirteen felt that the figures were helpful and not too few or too many. Most (eight) felt that Squeak was easier than other programming languages, four felt that it was harder, and four felt that it had the same difficulty. Table 2 summarizes the rating results.

The rating responses to the Part 2 Questions are favorable. For the middle-school group, the averages indicate that the student understood the tasks and could complete them. The most favorable response was to question 5. Here all but one of the students agreed with a 9 or 10 rating and tried changing the code beyond the given instructions before completing the tutorial. The least favorable responses were to questions 10 and 12. Given the limited time available (2 hours) for just Tutorial 1, the students would hardly be ready to do much independent coding. The low response to question 2 may indicate that the tutorials need more adjustment to the age group. Although, the youngest student gave this question a 10.

For the home school group and the public high school group, most of the categories were quite favorable. These older students had less difficulty completing the tutorials and felt comfortable with the length, instructions, layout, etc. They were less likely to experiment with

the code during the tutorials and seemed to follow the given instructions more closely. After completing the tutorials, both groups then generally experimented with the environment. The high school group organized an impromptu competition for who could make the "coolest" design. These groups gave the tutorials a relatively high overall recommendation (question 12).

From the open-ended responses, the younger students did not see themselves as programmers, while the older students could think of ways to use programming in a career or for fun. (One of the home school students wanted to teach programming to his mother and another wanted to explore using Squeak for artwork.) The groups suggested that action or movie graphics should be used, that interactive features be added, and that more color should be used. One student emphasized that he understood Squeak Smalltalk better than Java after just one session of Squeak. The art teacher and the computer skills teacher were present during the Dent-Phelps R3 implementation. While they did not perform the tutorials, they were very interested in the program capabilities and asked that the software and tutorials be left for their examination. They noted that the experience brought out active interest for some of the less outgoing students.

Summary

The Squeak Smalltalk environment was used for pre-college education with students as young as ten years old. The student participants were guided through their programming experience with a series of five tutorials. The intent of the tutorials is to allow independent progress through the instructional material without the need for formal instruction or the presence of a teacher. Squeak Smalltalk is a fully functional programming environment and is well suited for young or inexperienced learners. The object-oriented graphical nature of the programming environment lessens the need for detailed syntax understanding. Also, interesting tasks can be completed with little training, such as the

drawing tool used in the first tutorial, similar to an "Etch-A-Sketch". These tasks allow students to make individual variations to their programs to reinforce learning and enjoyment.

The tutorials were developed to address basic programming literacy issues. Programming is a useful component in developing critical thinking and problem solving skills and in preparing for careers, especially STEM-related careers. The intent of the tutorials is to teach basic programming skills related to Squeak Smalltalk and also to develop an understanding of general programming concepts. To address the pre-college audience, the tutorials were tailored with regard to length, task size, repetition, etc.

Three preliminary implementations with middle-school-age and high-school-age students were done. The students are clearly capable of understanding and completing the tutorial tasks. Several students, especially the younger students, showed a strong interest in developing their own code as they "play." The older students tended to follow the instructions more closely and had less difficulty using the tutorials. The results of the assessments indicate a need for more color and perhaps graphical aids and have identified some points of confusion. The results indicate that the level of the tutorials is good for high school students, but that more adjustment would benefit the young students. In particular, the younger students seem to need more aids such as graphics showing example screen shots and programming optional paths. Sidebars on applications and careers may be beneficial as well. The next version of the tutorials will incorporate related changes. Also, additional tutorials are planned to address more advanced concepts and to allow students to create a screen saver.

The Squeak Smalltalk software is available[11]for download at www.squeak.org and the tutorials and evaluation form are available at www.hawcenter.org/squeak. html.[12] Any groups using the tutorials are asked to assess the material with the evaluation form. As shown in Figure 6, the associated

Readme.pdf file gives version information, notes the supporting operating systems (Linux, Windows, and Macintosh), and gives instructions on general use of the tutorials.
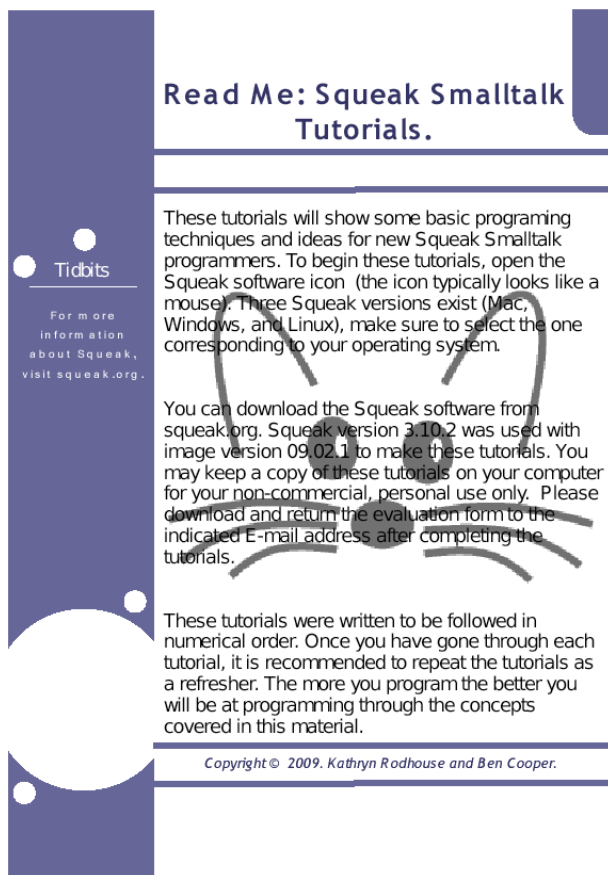

Figure 6: Readme.pdf File.

Acknowledgements

Assistance from the Dent-Phelps R3 School in Salem, Missouri, the Park Hill South School in Kansas City, Missouri, and the Hawthorne Center for Innovation in Inland Empire, California is appreciated.

Bibliography

1. J. Streeter, *Immigrant Scientists, Engineers, and Technicians:1993* (National Science Foundation, 2006).

2. National Science Board, "America's Pressing Challenge – Building A Stronger Foundation. A Companion to Science and Engineering Indicators – 2006," National Science Foundation (Jan. 2006).

3. Student Disdain for Science and Technology Threatens American Preeminence, Report Says," The Whitaker Foundation, 18 May 2004.

4. T. M. Swift and S.E. Watkins, "An Engineering Primer for Outreach to K-4 Education," *Journal of STEM Education*, Vol. V, Issue 3 and 4, 67-76, July-Dec. 2004.

5. BEST Robotics Inc. - Boosting Engineering Science and Technology. Available WWW: http://best.eng.auburn.edu.

6. S. Schneider, "Developing an Introductory Software Programming Course for Engineering Students," Proceedings of the 2005 American Society for Engineering Education Annual Conference and Exposition, American Society for Engineering Education, 2005.

7. K.P. Brannan and J.A. Murden, "From C++ to MathCad: Teaching an Introductory Programming Course with a Non-Traditional Programming Language," Proceedings of the 1998 American Society for Engineering Education Annual Conference and Exposition, American Society for Engineering Education, 1998.

8. American Computer Science League. ACSL. Available WWW: http://www.acsl.org/.

9. H. Said, "The Effect of Programming Learning Center on Students in First Year Programming Sequence," Proceedings of the 2004 American Society for Engineering Education Annual Conference and Exposition, American Society for Engineering Education, 2004.

10. A. Azemi, "Teaching Computer Programming Courses (Using the Internet) in a Computer Laboratory Environment," Proceedings of the 2002 American Society for Engineering Education Annual Conference and Exposition, American Society for Engineering Education, 2002.

11. Squeak Smalltalk. Squeak Oversight Board. Available WWW: www.squeak.org.

12. Squeak Smalltalk Tutorials. Hawthorne Center for Innovation. Available WWW: www.hawcenter.org/squeak.html.

Biographical Information

Kathryn N. Rodhouse will receive a B.S. in Computer Engineering degree with minors in Mathematics and Computer Science from Missouri University of Science and Technology in 2011. She plans to begin a M.S. program in computer engineering. She received a 2010 IEC William L. Everitt Student Award of Excellence at Missouri S&T and is a member of IEEE and Eta Kappa Nu. Her research interests include programming, image processing, and computational intelligence.

Benjamin Cooper is CTO/Managing Partner of Savant, LLC. He is an entrepreneur with experience in several companies. He attended Emory University and the University of California, San Diego.

Steve E. Watkins received his PhD from the University of Texas – Austin in Electrical Engineering in 1989. He holds M.S.E.E. and B.S.E.E. degrees from the University of Missouri-Rolla. He is currently a Professor at Missouri University of Science and Technology (formerly the University of Missouri-Rolla) and Director of the Applied Optics Laboratory. He is a member of IEEE (senior member), SPIE (senior member), ASEE, and Eta Kappa Nu. His research interests include optical sensing, smart system applications, and engineering education.

ASEE MEMBERS

How To Join Computers in Education Division (CoED)

1) Check ASEE annual dues statement for CoED Membership and add $7.00 to ASEE dues payment.

2) Complete this form and send to American Society for Engineering Education, 1818 N. Street, N.W., Suite 600, Washington, DC 20036.

I wish to join CoED. Enclosed is my check for $7.00 for annual membership (make check payable to ASEE).

PLEASE PRINT

NAME: _____

MAILING ADDRESS: _____

CITY: _____

STATE: _____

ZIP CODE: _____