AC 2008-318: STUDENT ATTITUDES TOWARDS THE USE OF GRAPHICAL PROGRAMMING LANGUAGES

Jeremy Garrett, Virginia Polytechnic Institute and State University

Jeremy Garrett is currently working on his Ph.D. in Curriculum and Instruction, with a specialization in Integrative S.T.E.M. Education, at Virginia Tech. His doctoral research, which he has recently begun, is on college freshmen-level engineering design curriculum. He has an M.S. in Applied and Industrial Physics from Virginia Tech, and a B.S. in Physics from Western (North) Carolina University. He has been teaching freshmen and sophomore general engineering courses for the last four years (some years as a lead teacher / instructor and some years as an assistant / GTA). Prior to that, he worked, for approximately two years, doing a combination of computer programming (primarily C++ and LabVIEW) and engineering research (fiber optic sensor design and testing as well as automotive adhesive testing).

Thomas Walker, Virginia Polytechnic Institute and State University

Tom Walker is an associate professor in the Engineering Education Department at Virginia Tech. His research interests are in the areas of active and collaborative learning, both synchronous and asynchronous in the engineering learning space, educational technologies, distance-learning, and object-oriented engineering design.

Student Attitudes towards the Use of Graphical Programming Languages In an Introductory Engineering Course

Abstract

In the fall and spring of 2007 freshmen CS, CPE, and EE students at Virginia Tech had the unique experience of working with five or six programming languages, all within one year and all for the purpose of developing fundamental programming skills. One of those languages was purely educational in nature (Alice), three were traditional and text-based (C++, Java, and MATLAB™ script), but two were unique graphical languages (RAPTOR™ and LabVIEW™). This paper briefly describes how teaching with graphical programming languages is consistent with the learning theories of constructivism and multiple intelligences. This paper also describes how a survey was used to take advantage of this unique opportunity to measure freshman student perceptions of relevance, general attitude, and recommendations for further use of each of these six programming languages. This paper concludes by describing the results of that survey and by discussing some implications of the results.

Keywords: freshman, graphical programming, computer programming, LabVIEW, RAPTOR

Introduction

In the fall of 2006 and spring of 2007 freshmen engineering students at Virginia Tech intending to enter into CS, CPE, and EE majors had the unique experience of working with five or six different programming languages, all for the purpose of developing fundamental programming skills. Of those five languages one was purely educational in nature (Alice), three were traditional text-based programming languages (C++, and MATLAB™ script programming), but two were unique graphical programming languages (Raptor and LabVIEW™). A few of the students also worked with Java, which is a traditional text-based language similar to C++. That unusual circumstance provided a rare opportunity to probe student attitudes towards the use of graphical programming languages in introductory programming courses, and to compare those attitudes against their attitudes towards both a purely educational language and traditional text-based languages in those same environments. In order to take advantage of that opportunity a survey was developed and implemented at the end of the spring 2007 semester. The survey asked the students to answer a common set of questions, eight questions for each of the six languages. Those questions included perceptions of relevance and perceptions of effects on self-confidence (also known as "self-efficacy"). The survey also asked the students whether, or not, they would recommend each programming language for use with future students. Although the surveys were anonymous, standard demographic data was requested, and that has allowed simple comparisons to be made not only between programming languages but also to compare the attitudes of women and minorities to those of white males (for this study, the responses of women and minorities were combined during the statistical analysis). It should be noted that the use of the terms "LabVIEW" and "MATLAB" refer to the registered trademarks for commercial products developed by National Instruments and MathWorks, respectively.

Background

During the 2006-2007 school year, two of the introductory engineering courses began, but did not complete, a process of transitioning from Alice to LabVIEW. Because of those reforms, the existing efforts to expose students to multiple languages (in an effort to spend additional time on critical, foundational concepts, and to enhance the transfer of those important concepts) resulted in the unique situation where most students were exposed to five or six languages within one year, rather than more typical two to four (C++ or Java, flowcharting with or without Raptor, Alice or LabVIEW, and sometimes MatLab Script). The key term here is "expose" and, with the exception of C++ and Java, that was all that occurred and that exposure took place in a comparatively active learning environment.

Before continuing, it is important to understand what the various programming languages are, the conditions under which the students were exposed to them, and why those particular programming languages are being used in introductory programming / problem-solving courses. With only a few exceptions, all of these students were required to study and use the Alice programming language for three weeks during their first semester of college in a general engineering course that is required, for all freshmen at Virginia Tech. The Alice programming language (which is named after the book *Alice in Wonderland*) is a fairly unique programming language, in that the source code is semi-traditional and text-based, but the output of the program is a three-dimensional, videogame-like animation, which readily lends itself to story telling[3]. Some educators and researchers believe that students who use the Alice programming language find it inherently fun and motivating because of the videogame-like output[7]. Research also shows that female students are often particularly motivated by Alice's story telling capabilities[7].

During that first year, nearly all of the responding students also took a semester long course on the C++ programming language, which generally occurred during their second semester (when this study was conducted). Although C++ supports, and is typically used with, object-oriented programming, it fully supports "structured programming" and other non-object-oriented approaches. With some additional libraries, C++ can be used to create graphics, and with enough work, it can even be used to create videogames, but those capabilities are generally considered to be "advanced topics" and are thus typically excluded from introductory C++ courses. As a result, some students might find working with C++ dull and boring. On the other hand, the traditional nature of C++ means that it is used by a high percentage of professional software development companies. Because of that, some instructors believe that students are motivated to use C++ because of its obvious relevance to future courses and future job requirements.

When this study was designed the authors believed that some of the students had taken a Java class instead of the C++ course, but the survey indicates that only a few students took a class in Java, and that many of those did so during their senior year of high school rather than at Virginia Tech. Java is like C++ in many ways; they are both text-based and commonly used by professional software developers. Unlike C++, Java is limited to object-oriented programming (as opposed to "structured programming"), and as a result, many students find Java to be a difficult first programming language. On the other hand, Java's ability to run on nearly any hardware and to control sophisticated web pages may provide students with higher levels of perceived relevance.

Unlike C++, students were only exposed to MATLAB script programming for six weeks during their second semester engineering course; which, unlike their first semester course, was specifically for future electrical engineering, computer engineering, and computer science students. Like C++ and Java, MATLAB script is a text-based language, but unlike C++ and Java, it is not a general-purpose programming language. Instead, it is used to automate the features of MATLAB. Although MATLAB is generally thought of as a tool for handling mathematics and matrix-based problems (which is how it earned its name, a contraction for Matrix Lab), these students were asked to use MATLAB and MATLAB script to manipulate audio files and image files which are stored digitally as large matrices. It was believed by the designers of that course (two of which are the authors of this document), that using MATLAB script in that way would help the students develop skills and knowledge that they could transfer to other situations and other program courses, while simultaneously demystifying the audio and video capabilities of personal computers.

LabVIEW, produced by National Instruments, was the first of two graphical programming languages that these students used. Unlike traditional text-based languages, LabVIEW is fully graphical in nature, and does not rely on a linear flow model. Instead, data flows the way that water and electricity do – data paths can fork, and merge, and can be either parallel or in series depending on how they are "wired" together. As a result, learning theories suggest that use of this language in introductory courses has many benefits. First, the theory of multiple intelligences states that there actually are multiple, separate types of human intelligence (bodily-kinesthetic, interpersonal, linguistic, logical-mathematical, naturalistic, intrapersonal, spatial, and musical, as opposed to a single type of intelligence described by "IQ") and that education needs to be tailored to the needs of those students. For example, the theory of multiple intelligences states that some students will benefit more from classroom examples that are visual, while others respond well to verbal (spoken) input, others to music, and so on[5]. Although this theory is controversial, studies have concluded that engineers and engineering students generally do prefer visual learning environments[4,6]. Second, the fact that LabVIEW builds only off of skills and knowledge that the students should have already developed in physical science classes, means they do not need specialized prior knowledge before they can proceed with learning computer programming skills (unlike Java, which requires prior knowledge of abstract class objects). Similarly, the water flow / electric flow model used in LabVIEW reinforces skills that these students are required to use in their subsequent electrical engineering courses. By building off of and further developing skills used in other environments, LabVIEW use promotes deep understanding of transferable knowledge, and its use is thus supported by constructivism and other learning theories.

The other graphical programming language that the students used is called "RAPTOR." Like Alice, RAPTOR was designed specifically to be an educational language. Unlike Alice, it is a graphical programming language based on flow charts[1]. As a result, its use in introductory programming courses has many of the same benefits as LabVIEW use, including being well suited for visual learners. At Virginia Tech, it was used in the students' second semester, after the students had already learned about flowcharts during their more general, first semester engineering course. In that way, the use of Raptor built on students' existing prior knowledge of flowcharts[2]. The students involved in this study were only required to use RAPTOR for two weeks, but during that time, they completed programming assignments that explicitly connected their RAPTOR flowcharts to MATLAB script, with the idea that this would promote the transfer

of knowledge from flowcharts to traditional text-based languages. In that way, its use is also supported by constructivism and other learning theories.

Problem Statement and Research Questions

As described, the use of visual / graphical programming languages would appear to have more benefit than the use of traditional text-based languages in introductory programming courses. However, as cognitive psychology teaches, student attitudes are critical in the creation of successful learning environments[8]. As a result, the problem statement for this study was, "What are student attitudes towards the use of graphical programming languages in introductory computer programming courses?" Within that question are several specific research questions:

I. "Do these students believe that their use of graphical programming languages increased their self-confidence in computer programming more or less than their use of text-based languages?"

II. "Is the students' general attitude towards the use of graphical languages higher or lower than their general attitude toward traditional languages?"

III. "Is their perception of the relevance of graphical languages higher or lower than their perception of the relevance of traditional languages?"

IV. "Is their perception of learning gains from the use of graphical languages higher or lower than their perception of learning gains from the use of traditional languages?"

Development Of The Survey Instrument

A survey was developed and later implemented in order to address the questions described above. To ensure content validity and an effective style, the rough draft version of the survey was presented to several experienced research faculty, some in Virginia Tech's Engineering Education department, and others in Virginia Tech's department of Educational Research and Evaluation. With that input, three versions of the survey were generated, each covering the programming languages in a different order, in order to equally distribute the negative effects of "tester fatigue" across the six languages being studied.

The following survey questions were used with each programming language. For this particular sample, students were asked about their perceptions of their use of the Alice programming language. In order to poll their perceptions of the other languages, these exact same questions were used, except that "Alice" was replaced with the name of the other languages.

Research Question:	Survey Questions:
I. "Do these students believe that their use of graphical programming languages increased their self-confidence in computer programming more or less than their use of text-based languages?"	1. Using the Alice programming language helped increase my self-confidence in the area of computer programming. (1) Strongly disagree (2) Disagree (3) Neither agree nor disagree (4) Agree

	(5) Strongly agree
II. "Is the students' general attitude towards the use of graphical languages higher or lower than their general attitude toward traditional languages?"	2. I would like to have additional programming activities using the Alice programming language. 3. I would prefer to use another language during my introductory engineering courses. 4. I would recommend additional use of this language in future introductory engineering courses. (1) Strongly disagree (2) Disagree (3) Neither agree nor disagree (4) Agree (5) Strongly agree
III. "Is their perception of the relevance of graphical languages higher or lower than their perception of the relevance of traditional languages?"	5. The Alice programming language does not seem related to my future studies in engineering. (1) Strongly disagree (2) Disagree (3) Neither agree nor disagree (4) Agree (5) Strongly agree
IV. "Is their perception of learning gains from the use of graphical languages higher or lower than their perception of learning gains from the use of traditional languages?"	6. I feel that my gains in computer programming, as a result of using the Alice programming language, were (1) Very high (2) High (3) Average (4) Low (5) Very low

In order to measure the amount of experience that each respondent had with each language, two additional questions were asked just prior to those shown above.

1. My first use of the Alice programming language was:
   (1) before the 9th grade (2) in the 9th grade (3) in the 10th grade
   (4) in the 11th grade (5) in the 12th grade (6) as a college freshman
   (7) as a college sophomore (8) as a college junior
   (9) I have not used this programming language

If you have **NOT** used this language, make sure that you have selected option #9, and then skip

the rest of this section.

2. I have used the Alice programming language outside of my officially assigned course work (such as EngE 1104 labs). (Note: Using it as an optional method of completing an assignment – such as using it to create product design visuals in the lecture/project portion of EngE 1104 – does count as using this language outside of officially assigned course work.)
   (1) true (2) false

Limitations

The research plan called for the survey to be administered during a standard class meeting. Although completing the survey was optional, the expectation was that this procedure would gather responses from nearly all of the approximately 280 students. Unfortunately, the tragic events at Virginia Tech on April 16, 2007 occurred shortly before the survey was to be administered. To allow for student recovery from that traumatic event, all classes became optional during the last three weeks of the school year. To address that issue, students remaining on campus were asked to voluntarily come back to class and fill out the survey for this study. This resulted in eighty-six completed surveys.

Another substantial limitation was the unequal length of exposure to the programming languages in question. In general the students only used Raptor, LabVIEW, and Alice for two or three weeks, and MatLab Script for six weeks, but nearly all of the students had either completed a full semester long course on C++ programming, or were currently about to finish such a course. Java exposure was unique in that the students typically had completed a lengthy course, or had not been exposed to the language at all.

Delimitations

Some of the survey results were excluded from the analysis because this study was interested only in the typical first-year, second-semester engineering student. Additional data points were removed due to incorrectly completed surveys forms, such as those that reported negative years of experience with a language (they reported their first use to be at a grade level higher than their current grade level). A few more were removed because they simply reported no use of a language but then responded to questions about that language. Responses from students who described themselves as non-freshmen / non-first-year students, non-engineering students, were also removed. Similarly, students who were not at least eighteen years old were removed. Thus, the data was filtered and sixty-eight valid data points remained.

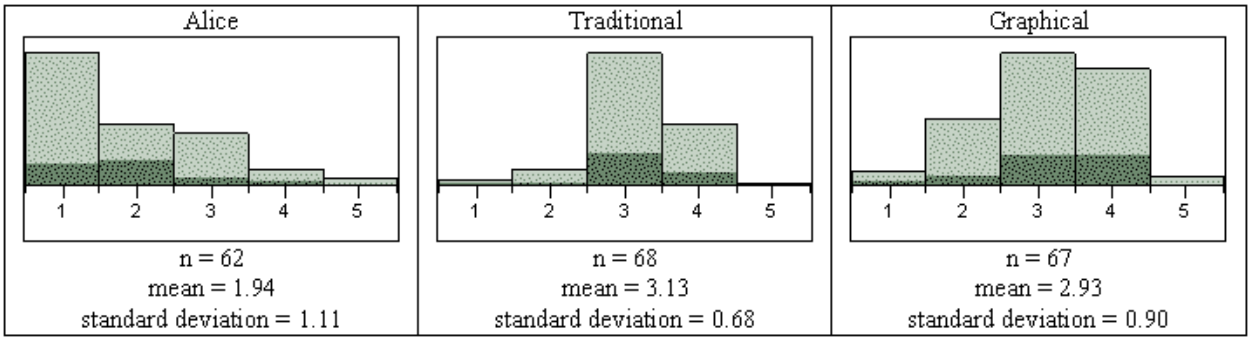
Data

Histograms of students' responses were generated for each survey question. In the case of the first research question, no special calculations were required. For the second research question, the results of survey question number three were reversed (a "1" became a "5" and vice versa), and then averaged with the results of questions two and four. For the third and fourth research questions, the results of the relevant questions (numbers five and six, respectively) were simply reversed. Then the responses to those questions about C++, Java, and MatLab Script were

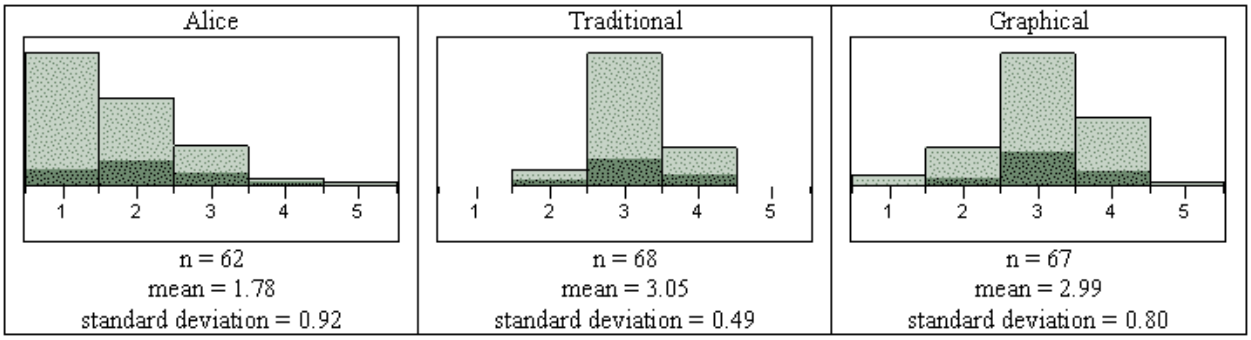
averaged together to produce mean responses about "traditional, text-based languages." Similarly the responses to those questions about LabVIEW and Raptor were averaged together to produce mean responses concerning "visual / graphical programming languages." Alice however was treated as its own category.

In order to gain some insight into how the use of these languages might or might not be compatible with a diverse student population, data from the twenty-seven students who were either female or a member of a minority racial group have been highlighted in the following graphs by making the data from those students a darker color. Because some students were not exposed to all of the languages (especially Java) and because some of the respondents did not answer every question on the survey, the number of data points for each programming language was unique. With the exception of Java (which had thirty-nine complete and usable responses), and C++ (which had forty-eight); the number of complete, usable responses for each language was between sixty-two and sixty-five. The number of responses for traditional languages and graphical languages is slightly higher than sixty-five, because the incomplete responses (due to both lack of experience with a given language and simple lack of completing the survey) were distributed across all of the programming languages rather than isolated to a single language or to a single language category.

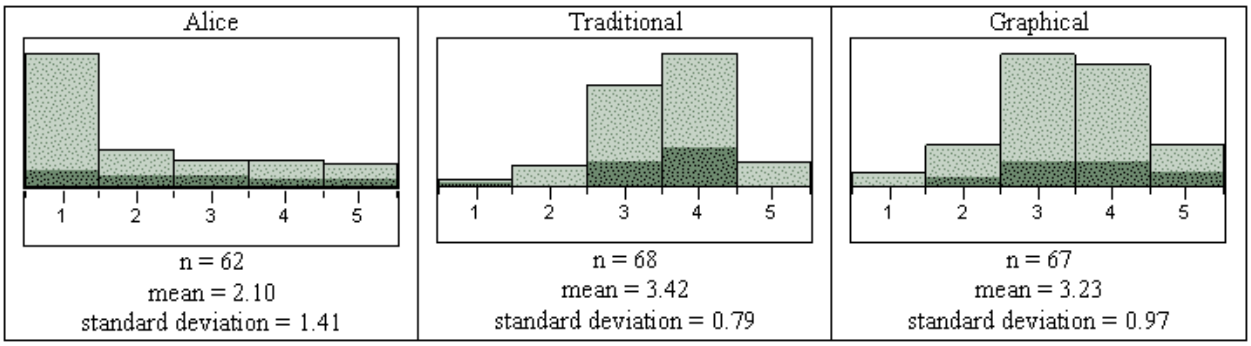
Research Question I – "Do these students believe that their use of graphical programming languages increased their self-confidence in computer programming more or less than their use of text-based languages?"



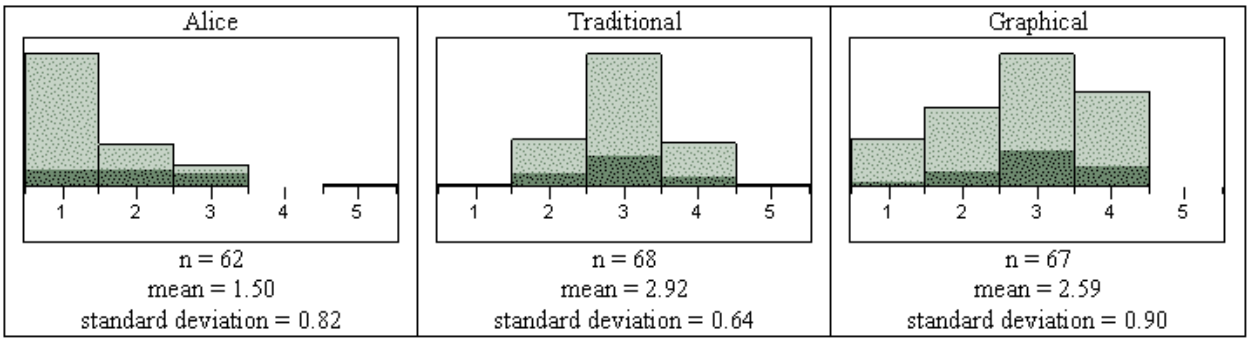
Research Question II – "Is the students' general attitude towards the use of graphical languages higher or lower than their general attitude toward traditional languages?"



Research Question III – "Is their perception of the relevance of graphical languages higher or lower than their perception of the relevance of traditional languages?"



Research Question IV – "Is their perception of learning gains from the use of graphical languages higher or lower than their perception of learning gains from the use of traditional languages?"



Data Analysis

Before continuing with an analysis of the histograms, it is important to examine the internal reliability of the data. One of the most common measures of internal reliability is "Cronbach's Alpha," which measures the tendency for responses to individual items to go up while the overall average goes up. Although this system has a maximum potential value of 1.0, at which point all survey responses move up and down together, a value of 0.70 or higher is typically considered satisfactory. Completing such a calculation for this data yielded a value of 0.85, indicating the data has a satisfactory level of internal reliability and that the data analysis can be continued.

First, it should be noted that part of the reason for the smaller standard deviations found in the data on traditional languages is caused by averaging over three separate languages. Although not shown, the standard deviations of the individual traditional languages and individual graphical languages were approximately equal and all slightly smaller than the corresponding standard deviations for the data on the Alice programming language.

Second, note that the highlighted data points, which represent students who are either females or members of minority racial groups, are approximately evenly distributed in all of the histograms.

With all four research questions, two-tailed Student t-Tests were used to determine whether the responses concerning traditional languages was statistically above or below the responses concerning graphical programming languages. Using an alpha level of 0.10, produced the following results:

Research Question #	p-value	Reject the Null Hypothesis?
I	0.2015	No
II	0.6145	No
III	0.3031	No
IV	0.0176	Yes

Conclusions

In spite of the limitations, the overall trends within the data do provide answers to the original research questions. The first question was, "Do these students believe that their use of graphical programming languages increased their self-confidence in computer programming more or less than their use of text-based languages?" The data indicates that students believed that even their short exposure to graphical programming languages produced the same amount of increase in self-confidence (or self-efficacy) as their use of traditional, text-based languages.

The second research question was, "Is the students' general attitude towards the use of graphical languages higher or lower than their general attitude toward traditional languages?" Statistically the general attitude of these particular students towards the use of graphical languages is the same as their general attitude towards the use of traditional, text-based languages.

The third research question was, "Is their perception of the relevance of graphical languages higher or lower than their perception of the relevance of traditional languages?" Students perceived graphical languages to have the same relevance towards their future studies as traditional languages.

The fourth, and final, research question was, "Is their perception of learning gains from the use of graphical languages higher or lower than their perception of learning gains from the use of traditional languages?" Unlike the previous research questions, the data produced a p-value of only 0.0176 (or 1.76%), meaning that the null hypothesis must be rejected. That means that students perceived higher total learning gains from their use of traditional languages than from their use of graphical languages. However, this is consistent with the amount and type of exposure the students had with the various languages – they used graphical languages for only two or three weeks each, focusing almost entirely on introductory concepts, but nearly all of these same students spent three or more months working with C++ while focusing on a variety of concepts.

Implications

Several implications arise when analyzing the answers to the four research questions together. First, even though these students were only briefly exposed to graphical programming languages their overall attitudes towards those languages were similar to their attitudes concerning traditional languages Although further studies would be required for verification, these results

suggest that graphical programming languages correctly build off the students' prior knowledge, and suit their learning styles (believed to be primarily visual). That in turn suggests that additional use of graphical programming languages would be beneficial to freshmen electrical engineering, computer engineering, and computer science students at Virginia Tech during their introductory programming courses. Unfortunately the tragedy at Virginia Tech on April 16, 2007 limited the number of respondents to not only a smaller than expected sample set but also limited them to the sub-set of the overall population that wished to return to class after the tragedy. As a result, a follow-up study is needed to expand the results of this study onto the entire population of electrical engineering, computer engineering, and computer science freshmen involved in introductory programming courses.

References

1. Carlisle, M. C. Welcome to the Raptor home page. Retrieved November 19, 2007 from http://raptor.martincarlisle.com/
2. Carlisle, M. C., T. A. Wilson, J. W. Humphries, and S. M. Hadfield, "RAPTOR: A Visual Programming Environment For Teaching Algorithmic Problem Solving," Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, ACM Press, 2005, 176-180.
3. Carnegie Mellon University, Alice: An Educational Software that Teaches Students Computer Programming in a 3D Environment, Retrieved November 19, 2007 from http://www.alice.org/
4. Felder, Richard M., "Learning and Teaching Styles in Engineering Education," Engineering Education, ASEE, 1988, 78(7), 674–681
5. Gardner, Howard, Multiple Intelligences after Twenty Years, Retrieved November 19, 2007, from http://pzweb.harvard.edu/PIs/HG.htm
6. James-Gordon, Yvette and Jay Bal, "Learning Style Preferences of Engineers in Automotive Design," Journal of Workplace Learning, 2001, 13(6), 239-245.
7. Kelleher, C. and R. Pausch, "Lowering the Barriers to Programming: A Survey of Programming Environments and Languages for Novice Programmers," ACM Surveys, June 2005, Retrieved November 19, 2007 from http://www.cs.cmu.edu/~caitlin/research.htm
8. National Research Council, How People Learn: Brain, Mind, Experience and School, ed. J.D. Bransford, A.L. Brown, and R.R. Cocking, Washington, DC, National Academy Press, Retrieved November 19, 2007 from http://books.nap.edu/books/0309070368/html/index.html