Incorporation of a 3D Interactive Graphics Programming Language into an Introductory Engineering Course

Jason Snook[1], Vinod Lohani[2], Jenny Lo[2], Kishore Sirvole[3], Jennifer Mullins[4], Jeff Kaeli[5], Hayden Griffin[2]

[1]Department of Computer Science, [2]Department of Engineering Education, [3]Department of Civil and Environmental Engineering, [4]Department of Curriculum and Instruction, [5]Department of Mechanical Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0106 USA

Abstract

Details of introduction of a 3D interactive graphics programming language (i.e., Alice) into an introductory engineering course (called "Engineering Exploration") at Virginia Tech are presented. The Alice system, provided freely (www.alice.org) as a public service by Carnegie Mellon University, provides a completely new approach to learning programming concepts. This is the FIRST large scale deployment of Alice (1260 engineering freshmen learnt it in fall 2004) in an introductory engineering course. One particularly challenging aspect of this was implementing mathematics into the Alice problem specifications. Two examples of simulating the Solar System and projectile motion of an object using Alice are briefly discussed. Qualitative analysis of instructor and student experiences are discussed along with quantitative survey results to measure the relative success of this initial endeavor.

Introduction

Virginia Tech's College of Engineering (COE) is the sixth largest US engineering program in terms of bachelor's degrees awarded in 2002[1]. All freshman engineering students at Virginia Tech undergo a common first year General Engineering (GE) curriculum developed by the Department of Engineering Education (EngE). The GE curriculum is undergoing major changes primarily due to two reasons: i) Recent addition of Computer Science(CS) into the COE, ii) More emphasis on engineering education research targeted at improving engineering pedagogy in the COE.

This paper will present the details of changes made to one of the introductory engineering courses (called "Engineering Exploration" or EngE 1024) in GE curriculum with particular reference to programming instruction, which constitutes about one-third of the course. In consultation with faculty members from CS and other engineering departments, the EngE faculty decided to introduce an object-oriented programming language called Alice into EngE 1024.

This is the FIRST large scale deployment of Alice (1250 engineering freshmen used it in fall 2004) as the introductory programming language of choice.

The rest of the paper is organized as follows: First, we briefly review the capabilities of Alice system and compare some programming concepts like control structures with a procedural programming language like MATLAB. Then, we'll briefly review related studies that have been reported in literature. Following this, we'll briefly introduce the EngE 1024 course. Development of Alice instruction material in EngE 1024 will be discussed next. One particularly challenging aspect of this was implementing mathematics into the problem specifications at the level freshman engineers should encounter in such a course. Two examples of simulating the Solar System and projectile motion of an object using Alice will be discussed in this regard. A number of surveys including a computer attitude survey, Alice pre-and post-test surveys were conducted during Fall semester to assess the effectiveness of Alice instruction. We'll briefly discuss our findings from the surveys before concluding the paper.

Alice Programming System

The Alice system, which is provided free of charge (www.alice.org) as a public service by Carnegie Mellon University (CMU), provides a completely new approach to learning programming concepts. Alice uses a 3D Interactive Graphics Programming Environment to teach the fundamental concepts of object-oriented programming. One of the major advantages of using Alice is the mitigation of syntax issues in lieu of teaching programming concepts. As those teaching programming can already attest to, present strategies for programming instruction require students to grasp not only the concept being presented but also the syntax necessary to implement said concept. For example, when teaching loops in Java or C, students must not only learn the concept of iteration but they must also learn how to code such an instruction. For example, students are presented an added hurdle when "do this 5 times" requires them to write "for (x=0; x<5; x++)" as part of a repetition control structure. In Alice, "do this five times" will be achieved as below (see Figure 1).

Figure 1 - Sample for loop in Alice

The 3D visualization that Alice utilizes also reinforces programming concepts by providing a

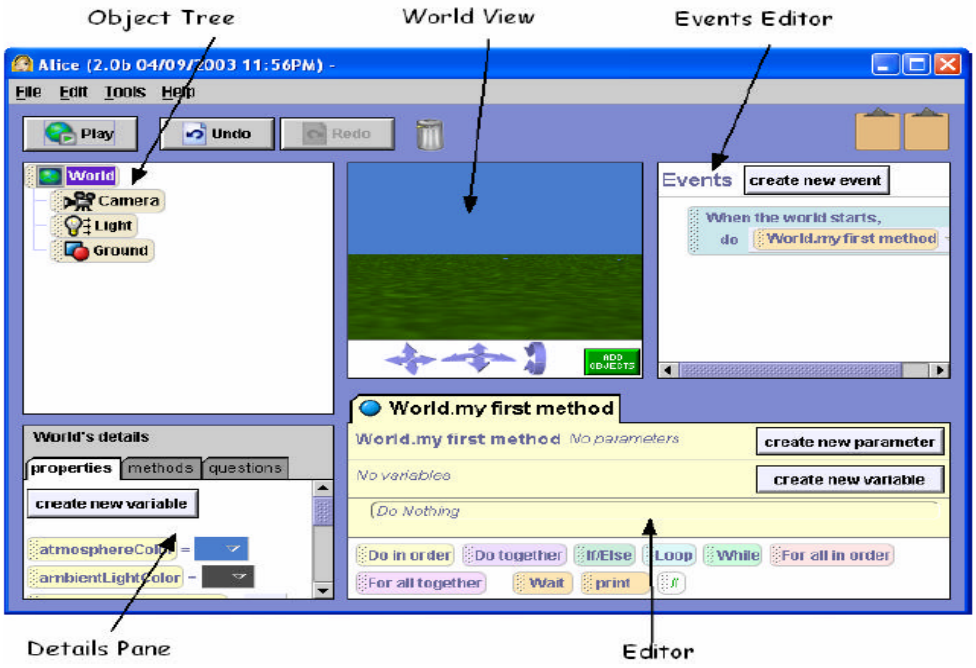


Figure 2: Alice User Interface

real-time representation of the code students are writing. The user interface, and even the name for that matter, was designed to take the fear out of programming for those new to the discipline. The basic user interface is composed of five major parts including the object tree, the world view, the events editor, the details pane, and the editor (see Figure 2 above). The object tree shows all of the objects currently implemented in the Alice world. The world view is the 3D visualization of the program (called a "world" in Alice) being implemented below in the editor. The details pane shows the attributes of a selected object similar to Visual Basic interface. The events editor allows users to attach commands to certain events, key strokes, or mouse actions. Since the Alice interface is certainly different from traditional programming interfaces, students with prior programming experience may face a certain amount of relearning to acquaint them with Alice. Those with less programming experience find the interface fairly intuitive to learn and use.

Related Work

Alice is being used as an introductory programming language in 28 different academic institutions including Virginia Tech[2]. However, at most institutions Alice is used in the introductory computer science courses. Authors are aware of only one institution (i.e., Bucknell University) where Alice was used in an introductory engineering course prior to its adoption in EngE 1024 at Virginia Tech. Alice is said to follow in the tradition of LOGO[3] and Karel which are similar educational simulation software packages used by novice programmers[4]. LOGO is a programming language useful mainly for children to learn geometry and related mathematical concepts. Karel was used to teach Pascal programming. Students can view the Karel world (environment) and can watch how the world changed with their program execution. Over the

past decade Karel has undergone several changes and is now known as Karel the Robot. Karel the Robot is used to teach Java to CS majors and math students at University of Waterloo[5]. Alice has been used in a core engineering course (i.e., ENGR 100) to teach a 3-week programming seminar at Bucknell University[6]. Approximately two-thirds of the students in this course had little or no prior programming experience. Despite their lack of programming experience, students were able to understand a wide range of fundamental programming concepts. In addition, students were introduced to even more advanced programming topics such as inheritance. In a survey distributed at the end of the course, a 5-point scale was used to solicit feedback. The average rating to a question regarding whether the Alice instruction had increased their interest in Compute Science was 3.73. When asked whether they would recommend this Alice seminar to other students, the average response was 4.05.

Alice has been attributed to improve retention and performance of incoming "at risk" computer science majors with little or no previous programming experience. In a study the effectiveness of Alice instruction in an introductory CS course was evaluated by collecting student grades and enrollment data over two years at two institutions namely, Saint Joseph's University and Ithaca College[7]. All the students enrolled for computer science major were subdivided into groups based on previous programming experience and math skills. The evaluation results show that at risk students (i.e., those who had no prior programming experience and weak in math skills) received significantly higher grades in CS1 than at risk students that did not participate in Alice. The overall result of the two years of data shows that at risk students who completed Alice course performed as well in CS1 as students who had prior programming experience. Use of a graphics library[8] to teach objects and methods prior to exposure to a complete programming is reported in a study. This library provides support for the use of an event –driven style of programming. The graphics library makes the code look simpler and the syntax in writing code is reduced.

Engineering Exploration Course Background

As mentioned earlier, freshmen engineering students enter as General Engineering students at Virginia Tech and take a set of common courses that must be successfully passed prior to moving into their degree-granting departments. This first semester engineering course is called Engineering Exploration (EngE 1024). EngE 1024 is designed to give freshman engineering students a fundamental understanding of the engineering profession and help them develop a set of tools that will be useful in their future engineering courses and careers. Over the past several years, the first-semester engineering course has evolved from a somewhat traditional problem solving, graphics, and programming course to a format that emphasizes early design and realization, collaborative learning, and highly interactive classroom environments [9-12]. One of learning objectives of EngE 1024 is to develop and implement algorithms that focus on object oriented approaches. As indicated earlier, due to joining of Computer Science department, computer science bound students are now required to enroll in EngE 1024 starting Fall 2004. This event transformed how programming is taught in EngE 1024, switching from MATLAB to an object-oriented language (Alice). The primary problem with the existing course was that instruction using MATLAB, which is inherently procedural, was viewed as an inappropriate first programming experience for computer science and computer engineering students, who will ultimately program in Java and C++, respectively. Moving directly into an object-oriented

programming (OOP) environment was desired [12-14]. Readers are encouraged to see a companion paper to see other changes that were introduced in the EngE 1024 course[15].

Training of Instructors and Alice instruction

Training of instructors/TAs

In all, nineteen instructors with a range of academic backgrounds, including Civil Engineering, Mechanical Engineering, Aerospace Engineering, Chemical Engineering, Civil Engineering, Biomedical Engineering, Industrial Systems & Engineering, Engineering Science and Mechanics, Metallurgical Engineering, and Electrical Engineering were involved in teaching EngE 1024 in Fall 2004. Much of the initial support and training in Alice came from the team at CMU led by Randy Pausch and two of his colleagues Wanda Dann from Ithaca College and Steve Cooper from St. Joseph's University. After initially hosting the lead author at CMU for 10 days, Drs. Pausch, Dann, and Cooper traveled to Blacksburg for a one day Alice workshop for the faculty. Faculty comments indicated this workshop was beneficial as an initial exposure to Alice getting them started as they picked up the technology. This one day workshop was followed up a month later by another one lead by our team for the rest of the faculty. From then on, faculty and TAs were encouraged to learn Alice via the tutorial included with the book and by reading the chapters we would be covering the first three weeks. Alice GTAs were made available for any questions or issues that came up and Friday training sessions for the faculty proceeded the weeks that Alice was used in the classroom. Throughout the semester, the Alice team at CMU and Drs. Dann and Cooper provided additional support via email and phone. Since most of the undergraduate TAs were hired after the workshops had happened, the textbook[16] was their primary source for Alice training. A total of 3 graduate teaching assistants (GTAs) and 12 undergraduate teaching assistants (UTAs) were hired from a variety of engineering disciplines. The involvement of undergraduates for instructor support and also the development of class exercises is an important priority for the department.

Alice teaching format

Alice instruction made up one-third of the 15-week EngE 1024 course. With two class meetings per week, a total of ten Alice lessons were prepared. In the first lesson, instructors delivered a short lecture on programming concepts demonstrated in Alice and then students were assigned a hands-on exercise to be completed before the end of class. In-class exercises gave students a chance to pair up with a classmate and try out what they learned in the first part of the lesson under the guidance of the instructor and one or two TAs. The second lesson consisted of a full lecture with a take home assignment given to students at the end of the class. Homework assignments were typically focused on concepts involving basic trigonometry and geometry or the simulation of a physical system (such as projectile motion).

Lecture topics were planned in a series of two blocks, a three week sequence followed by a two week sequence later in the semester (see Table 1). There was a heavy emphasis on basic

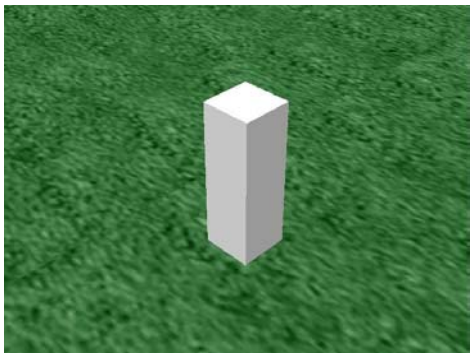
Table 1: Alice Instruction Format

Week	Topic (first class of the week)	Topic (second class of the week)
1	Introduction to the Alice interface and building a first world	Program design, implementation, testing, and commenting
2	Pure Functions and Expressions	If/Else statements and for loops
3	World and Class Level Methods	Parameters
4	Functions and Expressions (revisited), and While loops	While loops and variables
5	More Applications of Mathematics (Quadratic Equation)	Selected topics (code sharing, events, and sound)

programming skills such as program design and the use of control structures the first three weeks. The last two weeks focused more on the application of those concepts with some additional new concepts, like events and code sharing in Alice, covered. Throughout the first couple weeks of Alice, a number of instructors commented on how quickly the students were moving through the material. Whether the course content seemed "easy" or the pace "slow", it served as an indicator that students were learning the same material as before at a faster pace.

Examples of Alice HW Exercises and Semester Project

One particularly challenging aspect of adapting Alice to teach programming concepts to freshman engineering students was the emphasis on math concepts. Existing implementations of Alice for programming instruction focused solely on the programming concepts. For our implementation, we were equally concerned with reinforcing the mathematical concepts they should already know or were learning in the programs they were writing. So a concept such as geometry could be implemented rather easily (ex., "what is the volume of the cube on the screen"). But, at the risk of calling such a problem boring, it certainly does not take advantage of Alice's potential to not only visualize problems in 3D space but to have fun doing it. A different approach involving geometry involved moving any three objects of the students' choosing from any random point into a position exactly 3 meters away from the other two objects (i.e., the points of an equilateral triangle). Examples of these two potential problems can be seen and compared in figures 3a and 3b below.



Figures 3a and 3b – Example of a simple cube in 3D space

versus spatial relations of three objects in 3D space

Most of our success implementing math concepts into Alice programming came from the simulation of physical systems. One of the first homework assignments students were assigned was to develop a model of solar system in Alice using Kepler's law to determine the speed planets orbited around each other (see figure 4). Later on in the semester, students implemented projectile motion of an object using basic physics fundamentals. Using the 3D visualization capabilities of Alice, students were able to implement projectile motion and then experiment with different values for initial velocity and launch angle to see how the path of the projectile changed as a result. Figures 5a and 5b below show two projectiles at the apex of their arc using different initial velocities and angles.

Figure 4 – Solar System simulation using Keplar's Law of Planetary Motion

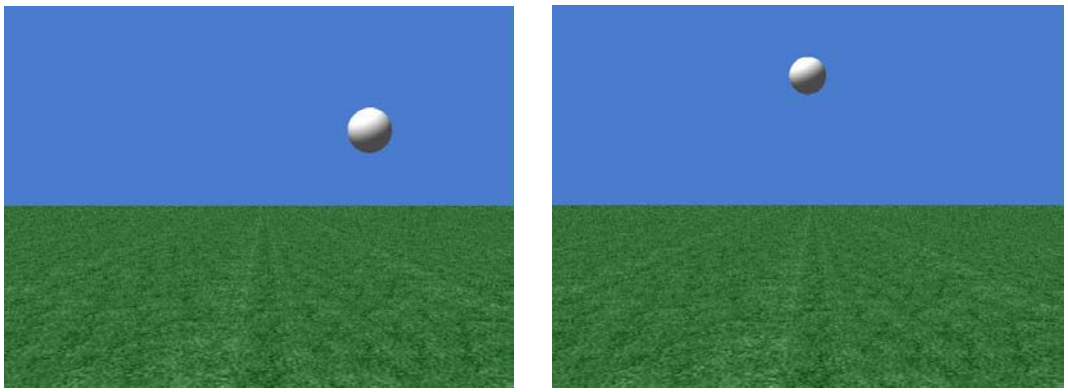


Figure 5a & 5b – projectile motion with different initial velocities and angles simulated in an Alice environment.

In addition to homework assignments, students were also assigned a semester project using Alice. Students were teamed up in groups of 5 or 6 and tasked to create a game ("rated 'E' for Everyone") that utilized the programming concepts they were exposed to during the course of the semester. This included at least one of each control structure covered (ex., for loops, if/else, etc.) and at least 5 world- and 5 class-level methods (In Alice, "world-level methods" involve multiple objects while "class-level methods" only involve the object it is contained within). Students were also required to include a mathematical concept in the project. Project requirements were left as open ended as possible to allow the students the maximum amount of room for creativity. Collaborative issues, such as dividing up programming tasks, sharing code, and integrating programming pieces, where also discussed in class and encouraged by the instructors. Approximately 300 projects were created as a result. Each student group was required to give a short in-class presentation of their project. Analysis of these projects is pending and will be included in a later article.

Alice bug reporting process

For this first semester of using Alice, a second type of TA was decided upon to assist students in the submission of bugs during class time. Testing TAs helped get students in the habit of submitting bugs so they would continue to do so during non-Alice weeks. Students were encouraged to submit bugs so they could be resolved ("Submit a bug, deal with it once, don't submit it and deal with it every time you use the program"). Over the course of the semester, this practice (and the prompt response of the Alice team at CMU) resulted in a fairly solid distribution of the program. A testing TA was present in every class that Alice was taught. A regular TA was also present during the first day of Alice weeks to assist students during the in-class exercises. This allowed the testing TA to focus on bugs while the second TA assisted students in difficulties attributed to the normal use of Alice. When there were no bugs, testing TAs were free to also help students with difficulties but bugs were their top priority.

Survey Results

Student Surveys

In the beginning of fall semester, a Computer Attitudes Survey[17] was administered to gather data on student attitudes towards computers and related technology. In addition, pre- and post-test Alice surveys[17] were conducted to assess effectiveness of Alice instruction. Of the 1267 first-year engineering students enrolled in EngE 1024 in fall 2004, about 40% had studied one or more programming languages including C++ (47%), Java (45%), HTML (45%), and Basic (40%) but only 0.2% had studied Alice previously. Nearly 89% had studied Calculus I or higher mathematics in high school and over 48% received a grade of A in Calculus. Over 75% had used programs like Word (99+%), Netscape or Explorer (92%), PowerPoint (87%), and Excel (78%). But 30% or fewer had taken a shop [industrial arts] class (29%) or a mechanical drawing or drafting class (27%) or enrolled in any pre-engineering courses (21%). Further, over 50% reported to have repaired an electronic device other than a computer or installed a hardware component inside a personal computer. More than 90% agreed ("definitely yes" or "somewhat yes") when asked: "Knowing how to work with computers will increase my job possibilities." Over 80% agreed when asked: "Using computers to write programs is a creative, logical pursuit of finding solutions to problems where the pieces can fit together in many, different, innovative ways" but less than 10% agreed when asked questions like: "I'm not the type to do well with computers," "I will do as little work with computers as possible," "I expect to have little use for computers in my life (3%)," "Learning about computers is a waste of time (2%)." The collected data has the potential to identify attitudes, which might predict success in engineering in general and particularly in those courses or course segments focused on computing.

Students were also required to take Dr. Richard M. Felder's Index of Learning Styles[18] survey online, which revealed that 85% of the students were considered visual learners and 66% were active learners. For those types of learners, it is likely that Alice enhanced the study of programming since (1) students could immediately see in 3D what was happening when Alice programs were run (2) students view an graphical representation of the code structure and (3) students had many hands on experiences with the software during class time.

Alice Instructors' Survey

Nineteen instructors participated in an anonymous survey that was administered after first six Alice lessons. The intent of the survey was to gather information that would help improve future Alice related lessons. Respondents expressed need for more mathematically rigorous Alice homework and in-class assignments, as well as simulations involving relevant physical models. There appeared to be a lack of unanimous agreement as to the accomplishment of the course objective to "develop and implement algorithms that focus on object-oriented approaches" using Alice based on six lessons that were completed until the time of survey. It may, however, be noted that majority of instructors had not taught object oriented programming previously. There were many insightful comments about improving the Alice instruction material. While there was a fair range of instructor experience in teaching computer programming concepts to freshman engineering students, a substantial representation felt that the students enjoyed learning Alice. Based on the feedback received from instructors, next four Alice lessons concentrated on simulation of projectile motion and code sharing procedure in Alice semester project. A more detailed report of instructor and student feedback will be included in a future article following a more comprehensive analysis of the data collected.

Conclusions

Many of the foundational concepts of object-oriented programming (i.e., classes, objects, methods) and of programming practice in general (design, control structures, methods and variables) were covered early and then applied together more extensively in the latter stages of the 5-week long Alice instruction. This structure worked well but faculty and student feedback suggests that concepts could have possibly been introduced at a slightly more accelerated pace leaving more room for application afterwards. Survey feedback above and more informal observations gave good indication that students found Alice fairly intuitive, although this seemed to be inversely proportional to prior programming knowledge and experience. The pace of programming instruction must be balanced by two factors: (1) maintaining a pace that challenges the students and keeps their interest and (2) setting a pace conducive to desired retention levels. Constant communication with the instructors and students is crucial in this endeavor.

Instructors' feedback and observation also spoke to the structure of class time during Alice weeks. Many found the short lecture and hands on exercise to be preferable to a full length lecture because it gave the students time to explore the concept experientially with their peers through the assigned exercises. In response to this and other such observations during other portions of the class, the department of Engineering Education at Virginia Tech will be reformatting its Engineering Exploration course to include a two-hour lab in place of one of the present lecture times. This will allow instructors to introduce concepts during the lecture and allow students to explore that concept hands on in the following lab. More details can be seen in a companion paper[15].

Over the course of the first semester using Alice, many ideas for math applications also came up. In addition to simulations of planetary and projectile motion, other future simulations will include pendulum motion, spring behavior, collision, and inertia. Greater inclusion of

mathematical concepts in 3D space will give engineering students experience with mathematical applications of programming and reinforce their observations through simulation of these forces in 3D space.

The implementation of Alice into the programming curriculum of EngE 1024 has provided a great deal of useful insight the authors hoped to present as best as possible in this paper. An oftentimes overlooked but nonetheless important element was the "fun factor" of Alice. Throughout the course students were observed "playing" with the program, creating Alice worlds and simulations just for the fun of it. Some students even tried to create rudimentary video games in their free time. As educators, we should jump for joy when we find students "playing" with a tool we have introduced to them, for it is that playtime that often leads to self-directed, experiential learning that goes beyond the course objectives and turns the students into active learners just for the sake of knowledge. And the more palatable the knowledge, the greater success we can expect in the education of students in the future.

Acknowledgement

The support provided by two NSF grants (i.e., award #s 0342000 & 0431779) is sincerely acknowledged. The guidance and support of Wanda Dann, Steve Cooper, Randy Pausch, and the entire Alice development team at Carnegie Mellon University is also greatly appreciated.

Bibliography

[1] Engineering Workforce Commission Report. 2002. "Engineering & Technology Degrees." Report from the American Association of Engineering Societies Inc.
[2] Personal Communication with Dr.Wanda Dann, Ithaca College, Ithaca College, Ithaca, New York.
[3] Papert, S., *MindStorms: Children, Computers, and Powerful Ideas.* New York: Basic Books, 1980
[4] Wanda Dann, Stephen Cooper, Randy Pausch: *Making the Connection: Programming With Animated Small World.* Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education. Helsinki, Finland, July, 2000.
[5] Byron Weber Becker: *Teaching CS1 with Karel the Robot in Java.* SIGCSE 2001 2/01 Charlotte, NC, USA
[6] Richard Zaccone, Stephen Cooper, Wanda Dann: *Using 3-D Animation programming in a core engineering course seminar.* 33rd ASEE/IEEE Frontiers in Education Conference, Boulder, CO,November 5-8, 2003.
[7] Barbara Moskal, Deborah Lurie, Stephen Cooper: *Evaluating the Effectiveness of a New Instructional Approach.* 35th SIGCSE technical symposium on Computer science education, Norfolk, Virginia, March 3-7, 2004.
[8] Kim B.Bruce, Andrea Danyluk, and Thomas Murtagh: A *Library to Support a Graphics –Based Object –First Approach to CS1.* 32nd SIGCSE technical symposium on Computer science education, Charlotte, NC, February 2004.
[9] Goff,R.M. and Gregg,M.H., "Redesign of a Freshman Engineering Program for the New Millenium," ASEE Southeastern Regional Conf, April 6-8, 1998, Orlando, FL.
[10] Goff, R.M. and Gregg, M.H. "Why Hands-on Design? A First Year Hands-on Design & Dissection Laboratory", 1998 Industrial Designers Society of America (IDSA) National Design Education Conference. Long Beach, CA September 21-23, 1998. Proceedings are on CD.
[11]York, S. C., "Providing early design/build opportunities to Freshman Engineering Students", *ASEE 2002 Annual Conference and Exposition,* Montréal, Quebec Canada, June 16-19, 2002
[12] Griffin, Jr, O. Hayden, Fox, E. A., Ribbens, C. J., Walker, T. D. L., Davis IV, N. J.,Goff, R. G., Lo, J. L., Lohani, V. K., Gregg, M. H., and Barnette, D., "Work in Progress – A Freshman Course for Engineering and

Computer Science Students." 34th ASEE/IEEE Frontiers In Education Conference, Savannah, GA, October 20-23, 2004.
[13] David J. Barnes & Michael Kölling, *Objects First with Java - A Practical Introduction using BlueJ*, Second Edition, Prentice Hall / Pearson Education, 2005.
[14] Cooper, S., Dann, W., and Pausch, R., "Teaching objects-first in introductory computer science," Technical Symposium on Computer Science Education, Proceedings of the 34th SIGCSE technical symposium on Computer science education, Reno, Navada, pp 191 – 195, 2003.
[15] J.L. Lo, Richard M. Goff, V.K. Lohani, T.D.L. Walker, T.W. Knott, and O.H. Griffin, Jr., "New Paradigm for Foundational Engineering Education", to be published in the Proceedings of the 2005 American Society for Engineering Education Annual Conference and Exposition, June 2005.
[16] Dann, W., Cooper, S., Pausch, R. *Learning to Program with Alice.* Beta Version, Pearson Prentice Hall, 2005.
[17] Moskal, B., Dann, W., and Cooper, S., 2004. Surveys on Student Background, Computer Attitudes, and Alice Concept Examination, Contact person: Wanda Dann, Ithaca College, New York.
[18] Felder R., Index of Learning Styles. http://www.ncsu.edu/felder-public/Learning_Styles.html

Biographies

JASON S. SNOOK and is a Ph.D. student in Computer Science at Virginia Tech. He is currently finishing up his dissertation involving statistical models of software use and adoption. He has been involved in all levels programming instruction for several years now and currently leads the Alice TA team at Virginia Tech in the implementation of this exciting new technology.

VINOD K. LOHANI is an associate professor in the Department of Engineering Education at Virginia Polytechnic Institute and State University (Virginia Tech). He received a Ph.D. in civil engineering from Virginia Tech in 1995. His areas of research include engineering education and hydrology & water resources.

JENNY L. LO is an assistant professor in the Department of Engineering Education in the College of Engineering at Virginia Tech. She received her Ph.D. in chemical engineering at Carnegie Mellon and her B.S. in chemical engineering at Tulane University.

KISHORE SIRVOLE is a M.S student in Civil Engineering department at Virginia Tech. He is currently doing research in transient analysis in water distribution systems. He is also a part of Alice Team at Virginia Tech.

JENNIFER MULLIN is currently a doctoral student in Technology Education with a bachelors degree in Mechanical Engineering. She is working with the newly formed Engineering Education Department at Virginia Tech developing instructional strategies for freshman engineering coursework which integrates contemporary cognitive theories. Her research interests include application of learning theory to engineering curricula, as well as exploring student's areas of interest in science and technology.

JEFF KAELI is a junior majoring in Mechanical Engineering at Virginia Tech. He has worked on the Alice Team as an undergraduate TA and in developing classroom exercises.

HAYDEN GRIFFIN is currently professor and head of the Department of Engineering Education at Virginia Tech. He holds BSME and MSME degrees from Texas Tech University and a Ph.D. in Engineering Mechanics from VPI&SU. He had 13 years of experience in industry and government laboratories prior to joining Virginia Tech in 1985. Prior to moving into his current position, he was associate dean for academic affairs in the College of Engineering.