"Inform, Experience, Implement" – Teaching an Intensive High School Summer Course

Ryan Layer, Mark Sherriff, and Luther Tychonievich
Department of Computer Science
University of Virginia
Charlottesville, Virginia 22904
Email: {rl6sf, sherriff, lat7h}@virginia.edu

***Abstract*—During the summer of 2011, twenty-four high school students participated in an intense, three-week computer science course at the University of Virginia. The course met for twenty-one three-hour sessions, thus encompassing more contact time than a standard college-level course. The course was structured in an "Inform, Experience, Implement" active-learning format: students were exposed to the history of a particular problem in context and participated in an active learning lesson regarding the topic before learning how to address the examined problem through programming. This structure helped integrate into the course best practices from experiential learning, kinesthetic outreach activities, and active learning pedagogy. Utilizing this three-part rotation curriculum achieved some important goals, including holding the interest of students during the summer for six hours a day and successfully motivating students who had no programming background.**

I. Introduction

Diversity continues to be a challenge in many Computer Science departments [1]–[5]. While work is ongoing in fostering diversity inside existing CS programs, many have noted that early exposure to computing, such as in middle school and high school, is essential in eliminating many of the stereotypes and misconceptions about computer science and thus helping to improve the numbers of incoming students into the discipline [6], [7]. However, many middle schools and high schools lack the resources and personnel needed to run such programs during the school year. To help fill this gap, numerous computing summer camps have cropped up at universities around the United States sponsored by various companies and organizations.

One of the main difficulties faced by computing summer camps is in creating a curriculum that can hold students attention for several hours a day, particularly during a time that they are accustomed to spending in more leisurely pursuits. Some summer camps minimize this difficulty by recruiting students that already have some familiarity and fascination with computing. While teaching such students in a concentrated learning environment is both exciting and rewarding, not every summer program has students that come in with that level of familiarity or drive. Indeed, these pre-motivated students are the ones who least need the early outreach of a summer camp in order to choose a computing field.

We embarked on creating a curriculum for students without prior interest in computing during the summer of 2011 at the University of Virginia. Our camp consisted of 24 fifteen-year old students from traditionally underrepresented demographic groups in computer science. All but two of these students had no programming experience whatsoever and mainly came to the program out of curiosity. Further, we were to have these students for six hours a day, separated into two three-hour blocks, over the course of three weeks. This equated to more contact hours than a standard three-credit college course.

We faced two main challenges with our summer experience: 1) how to hold a young student's attention during a traditional break time; and 2) how to cover enough material to make an impact without overwhelming the students.

In this paper, we describe the curriculum that we developed for an intensive summer computer science experience for high school students with no programming background. Our goals in this camp were twofold. First, we wanted to treat the camp as an outreach program, with the goal of building interest and excitement in the students. Second, we wanted to provide the students with practical programming skills and experience they could use in the years that would likely elapse before their next exposure to computing instruction. Thus, we decided to include the kinesthetic activities that have been shown effective in outreach, the student-focused educational techniques that have been shown effective in internalizing knowledge, and some degree of direct instruction as the students would not have the time to investigate course topics on their own before experiential instruction. We combined these elements in a curriculum that was based around the mnemonic of "Inform, Experience, Implement", where our instructional style progressed from contextual history lessons, through active learning experiences, and ending with programming implementation of various solutions. This methodology allowed us to integrate our various objectives in a direct and accessible way, both building interest and developing skills in the students. We found that keeping this mnemonic in mind through our teaching was an effective method for this age group and experience level combination.

II. Related Work

In this section, we describe related work in computer science summer camps and teaching computer science material in the context of the real world.

A. CS Summer Camps

There have been many studies that indicate computing summer camp programs can increase the interest of females and under-represented minorities in computer science (see, e.g., [8]–[21]) and that have discussed other aspects of summer camp design (see, e.g., [22]–[29]).

We note that there is a bredth of discussion on summer camps in educational venues in part because of the many different potential combinations of student age, experience level, demographics, and affluence, along with the wide varieties in contact hours with the students. For example, Miller et al. describe their one-week camp in Colorado as being targeted specifically for middle school students from low-income families [9]. Meanwhile, Carmichael created a day camp specifically for girls in the realm of video game development [15]. Camps range from one to several weeks, with students living on campus and off, and with contact hours ranging from one to seven per day. Ages range from ten to eighteen, with some camps specifically for girls, other for boys, and others still for students that are from underrepresented demographics in computer science.

Our goal is to add to the growing body of knowledge on how computer science educators can successfully run a summer experience for younger students. We feel that our experience is unique from the others in two distinct ways. First, our number of contact hours was greater than the other camps that we examined, totaling more total contact hours than a standard three-credit college course. We believe that our experiences with a camp of this scale can help other instructors tailor their own efforts effectively. Second, our students came from many different cultures and locations. We had students from across the country, not just local to Virginia, and all were from traditionally underrepresented demographics in computer science. Only two of our students had any programming experience prior to coming to the camp.

B. Teaching CS in Context

Teaching computer science in the context of computing problems is something that has been debated in CS education. The argument for contextualized CS education focuses on the idea of grabbing the interest of the student and using that hook to then introduce various computing topics. If the context is "good enough," then students are more likely to stick through a CS course and go on to a second one [30], [31]. The counter-argument states that students learn computing topics regarding that one context and that introductory students have a difficulty abstracting the concepts that they have learned to be able to apply them to other problems. Further, teaching the context takes up valuable lesson time, which might be spent on covering possibly more advanced programming topics that could have the same luring effect as the problem context [31].

Guzdial argues that if students don't learn the material in the first place, then they won't have the opportunity to transfer what they learned from one context to another [31]. Thus, if teaching in context helps students decide to stick with CS where without such context that would leave the course, then there is a net gain. The main idea behind this and other contextualized instruction arguments is that students want to know why what they are learning is relevant and useful [31], [32].

The desire to teach computing in context is almost axiomatic in summer camp and enrichment programs, with camps foccusing on such topics as robotics [11], games [15], [23], [24], [26], publishing [16], web development [20], [21], fashion [14], and creativity through computing [10], [29], [33] to name just a few. We likewise found this desire to understand relevance manifest with our group of high school students. With this age group and experience level, we quickly determined that if we did not provide them with context early on, attention and interest waned.

We had six hours a day with the students which allowed time to cover both the context and the programming in reasonable depth. Our "Inform, Experience, Implement" concept came specifically from the mentality that we had to sell the students on the problem first, have them experience the problem personally, and then learn how to solve the problem through programming.

Multiple studies have argued that mixing various media and motivated examples improve student involvement and learning [33]–[37]. These fit into broader trends in science education to engage students in the learning process through various problem-oriented and student-directed learning structures [38]–[42]. Recent studies have also observed that some activities that have been shown to engage students are not well suited to teaching computing at a deeper level [43], [44], suggesting there may be additional value in mixing instructional approaches.

The main challenge we faced was in devising a structure that would allow us to easily integrate teaching the context of problems, instructing students in core elements of programming, and utilizing kinesthetic, inquiry-based, and student-directed learning. We developed the mnemonic "Inform, Experience, Implement" as a tool to help guide our lesson development, providing a framework that aided our lesson planning to give students opportunities to engage through classroom instruction, learning activities, and programming experience.

III. The Course

In this section, we describe our curriculum and provide an example lesson that we used during the course.

A. Overview

The goals of the 2011 summer program at the University of Virginia were to instill excitement in students about computer science and to provide students with a solid foundation in a production-level programming language, Python. We selected the Python programming language because we believed it balanced accessibility and power. The interactive command shell allowed students to dive directly into programming without the added complexities of editors, interpreters, and environment setup. The students easily moved from typing

short commands in the shell to writing programs in text files as the commands became too long for comfortable shell use.

Our 2011 class had 24 rising high school sophomores and juniors from seven states. A large majority of the students identified themselves as having no or very little programming experience, a small portion had experience with HTML, and two students had a working knowledge of at least one programing language. For three weeks, our students lived in on-campus dorms, attended classes, went on field trips, and completed nightly homework assignments. Their typical day consisted of two three-hour classes and a two-hour study hall session. In total, students attended 21 three-hour-long classes, totaling more contact hours to a standard college course. In addition to class-based instruction and assignments, students worked in small groups on a final project that they presented to a representative from our corporate sponsor on the last day of the program.

B. Staff Composition

The course was taught primarily by one CS faculty member and two graduate students with significant teaching experience. Other CS faculty members and grad students were brought in for either a full three-hour session or a half session to cover a lesson in their particular research area. By splitting up the teaching and having several guests speak on their expertise area, we were able to avoid over-utilizing any particular instructor and kept the instruction fresh for the students. Guest instructors framed their lectures however they liked, with the main instructors of the program arranging the schedule such that the "Inform, Experience, Implement" rhythm was continued.

C. Course Content

One of the messages we wanted to convey to the students is that computing is a broad and enabling field that impacts every aspect of the modern world. To this end, we touched on many topics, including cryptography, GPS and mapping, the Internet, artificial intelligence, game programming, and scientific computing. Additionally, we had pairs of students team up to create projects to show off at the end of the course, and provided time for problem-based learning to accomplish that goal. A full outline of course topics can be found at http://stardock.cs.virginia.edu/lead.

We included the "Inform, Experience, Implement" approach within each topic, but not always in a straightforward sequence. For example, in game programming we used extreme programming to repeatedly iterate from experiencing a functional but incomplete game to discussing how it could be improved and implementing those improvements. Another example of adjusting the design is path planning, where we did some of the implementation in advance and used it to increase the learning during the subsequent experience activity.

IV. Example Lessons

To better illustrate how our mnemonic of "Inform, Experience, Implement" informed our course creation, we provide several example lessons from our camp. A full course schedule and outlines of other lessons may be also be found at http://stardock.cs.virginia.edu/lead.

A. Cryptography

After a couple basic introductory lessons, we began the "Inform, Experience, Implement" methodology by teaching cryptography.

The "Inform" lesson was organized around the history of cryptography. The first hour introduced the concepts of encryption and decryption, starting with classic ciphers such as Caesar, pigpen, and Vigenére. Students worked in teams to encode and decode simple messages provided by the instructor. In the second hour, the instructor described how modern computing has cracked classic ciphers through brute force, frequency analysis, and other similar techniques. Students then learned how cryptography has evolved, looking at modern schemes such as RSA. The third hour the students developed their own encryption schemes. Each pair of students was given a standard deck of playing cards. The pairs had to come up with their own encryption methodology, write down a detailed algorithm of how their scheme worked, and then encode their deck of cards to send a message to another pair. Thus, the "Inform" activity in this case actually ended with teaching a lesson about how to write basic algorithms for others to follow.

The "Experience" lesson began during the next morning's three-hour block. As the students had not yet seen most of the campus, a scavenger hunt was arranged for them to find various clues around the grounds. However, each clue was encrypted with a different scheme, some from the day before and some that they had not yet seen. This lesson both gave the students a well-needed active exercise to get their mind focused for the afternoon and emphasized examining and applying new algorithms in a repeated fashion.

During the afternoon's "Implement" block, students were asked about the encryption schemes they used. Several students noted that doing Caesar cipher shifts by hand was tedious and that they wished there was a way to automate the process. Similar sentiments were expressed with the other ciphers as well. This desire motivated our introduction of the concepts of strings and loops as we showed the students how to write programs that could encrypt and decrypt these cyphers more rapidly and more reliably than the students could do by hand.

B. GPS and Maps

Another good example of our curriculum pattern is a lesson set on GPS. The "Inform" lesson began with a history of celestial navigation using a sextant and worked up through modern GPS technology. Particular attention was paid to the impact software has had on navigation, including algorithms to deal with errors such as skew and jitter. Students also learned about the great circle distance formula used to find the shortest distance between two points on the surface of a sphere, and calculated several distances by hand.

For the "Experience" lesson, students used Andriod-powered smart phones to map buildings on grounds. Each

smart phone ran a simple custom application that continuously displayed the current GPS coordinates, had a button to add the current coordinates to a list, and another button to email the coordinate list to the user once the activity was completed. Students were divided into teams, and each team was given a list of buildings to map. While the list of buildings each group of students mapped were unique, each building was mapped by at least two groups. This organization allowed the full set of points to be combined into a larger map, and the duplicate points demonstrated issues with GPS coordinate gathering.

In the "Implement" lesson students learned to read data from a file, graph points, and implement custom functions. Prior to this point in the class, each values had been either hard-coded or read from the keyboard. The long list of coordinates, where each coordinate consisted of many characters, motivated the need for file input. Once the coordinates were read from the files, students learned to graph points on the screen and connect points to form shapes. The class combined coordinates from all groups to form a rough map of grounds. Finally the class wrote a function that implemented the great circle distance formula. Later on in the camp, students also used these coordinates when learning about optimal paths as we had them act out the traveling salesman problem by delivering "packages" to buildings on campus.

V. Discussion

As an instructional staff, we took several lessons from this experience.

A. Classroom Management

We found managing class time was challenging. A typical day in our course consisted of two three-hour sessions with an hour lunch break between sessions. In our experience, few students enjoy sitting through courses lasting more than an hour, and college instructors typically lack experience managing three-hour class periods. A partial solution was to split each three-hour session into two sessions separated by a short break. However, we found it difficult to make breaks long enough for the students to relax and take care of personal business while still short enough that the students didn't lose focus and initiate lengthy social activities. The most effective counter to this problem was having lessons that were more active and open ("Experience" lessons), which helped the students to feel that they were not trapped in the lab.

PowerPoint slides were not an effective teaching tool for this type of course. We first tried having slides that contained both definitions of various Python concepts and annotated examples of Python code for the students to type in and experience. While the students fully participated in typing in the code, there was a clear disconnect between the slides and the programs. Many students focused on entering the list of commands, not on understanding the commands and the relationship between commands and results.

Because of this disconnect, we switched to executing the commands along with the students. This switch gave us more flexibility to demonstrate the dynamics of each command and made it easier to field questions from the students. After the switch from slides to the command environment, we polled the class about the two styles, and the students overwhelmingly preferred the command environment.

One drawback of using the command environment instead of slides is the inability to annotate commands. Without annotations, all our explanations were verbal and students who were learning at a slightly different pace sometimes missed our explanations. While it may be possible to use both environments, we found that switching between slides and the command environment was cumbersome and disruptive.

B. Student Participation

The rotation between "Inform" lessons, "Experience" lessons, and "Implement" lessons had several positive effects. First, the students stayed engaged during the three-hours lessons. Putting problems in context and making the assignments meaningful did have a significant effect on the group. Knowing that an active learning session was coming up seemed to keep them more attentive during the lecture and lab sessions. By the time we got to an "Implement" session, students were eager to try out their own ideas on how to solve certain problems.

The students reported that game programming was their favorite topic. For their final projects, all but two of the teams created some form of game. Students indicated that the interactive nature of their projects played a large role in why it was interesting to them. Other interesting projects included a networked, chat-enabled version of Tic-Tac-Toe, a quiz application, and an interactive music creation system. The final project teams each gave a formal presentation of their projects to the camp's corporate sponsor on the last day of the program.

C. Results

The results of our camp experience can be evaluated from at least two perspectives. As instructors, we found that "Inform, Experience, Implement" greatly streamlined the process of creating lesson plans that incorporated kinesthetic activities, problem-based inquiry learning, topical instruction, and practical hands-on experience in many topics within computer science. Recalling the mnemonic allowed us to easily check if a particular lesson plan was likely to include each of these research-backed pedagogical and outreach elements.

The students also seemed to benefit from the course. Three months after the program completed, we sent informal surveys to the students asking about their experience. Results indicate we achieved our goals of motivating students to explore computer science and providing a foundation for further study. Selected results from the survey are below.

1) We asked students how comfortable there were with computer programming. On a scale from one (not comfortable at all) to seven (very comfortable), half of the respondents rated themselves as a three or less before the program; 75% of students rated themselves as a six or higher after the camp.

2) We asked students to rate how enthusiastic they felt about computing. On a scale from one (not enthusiastic at all) to seven (extremely enthusiastic), half of the respondents rated themselves as a four or less before the program, all picked at least a five and 63% of students chose seven after the camp.
3) We asked students if they are more likely to pursue computing in college or a career after the camp. On a scale from one (No!) to seven (Yes!), 88% of the students rate five or higher.

In addition to these statistics, each individual student rated themselves as more comfortable with and more enthusiastic about programming after the camp than they were before.

VI. Use in College Courses

After the success of the first instance of our course, we looked to find ways to incorporate our methodology into standard college classes. As one of the authors is the primary instructor for Introduction to Programming at the University of Virginia, we felt it best to pilot some of our lessons in this course. Introduction to Programming is a required course for all engineering students at the University of Virginia and is also taken by a large number of students from the College of Arts and Sciences. In a given year, over 1000 students take some version of our introductory course, with around 750 taking this particular version. We felt that this class would be a good fit for the technique due to its diverse population of student backgrounds and interests and the natural fit the methodology would have with a close lab format.

"Inform, Experience, Implement" lessons were conducted on encryption, GPS, and Python programming. For the most part, the lessons were well received. Many students anecdotally reported that they enjoyed doing things outside of the traditional classroom lecture format, particularly when the classroom has nearly 300 students in it. However, there was a vocal minority that disliked being asked to do something that required them to leave the classroom. Student performance on test questions on this material was, on average, similar to other question score distributions. However, as these topics had not been introduced into this class in any other fashion, we cannot draw any conclusions as to whether the technique itself improved learning. We plan to revamp existing course material to reflect the "Inform, Experience, Implement" methodology to make a better comparison in our future work.

VII. Future Work

As discussed in the previous section, we are branching out to utilize the "Inform, Experience, Implement" methodology in other courses and formats. Besides using the methodology in introductory college courses, we will use the methodology again in the summer of 2012 with the next iteration of our LEAD summer camp. This summer, however, we will be running two one-week courses, thus substantially changing the amount of time we have with the students. Further, our age group is dropping by a grade level, to rising ninth grade students. Our goal this summer is to evaluate the efficacy of the methodology when compressed into a tighter schedule. We plan to continue to do lessons on encryption, GPS, and Python, while introducing a new topic on robotics using Lego Mindstorm NXT kits.

Early results from the 2012 summer course are generally positive, with student-reported comfort with computing increasing from 2.7 to 5.0 on a scale of 1 to 7, with 7 being most comfortable. We plan to present more complete results from the 2012 camps in the future.

VIII. Conclusion

During the summer of 2011, twenty-four high school students participated in an intense, three-week computer science course at the University of Virginia. The course itself encompassed more contact hours than the standard college-level course. Due to the amount of time that students spent in class (six hours a day split into two three-hour sessions), we looked for novel ways to keep students' interest and enthusiasm high. We found that teaching CS in context for this specific demographic and experience level was highly effective. The course was structured in an "Inform, Experience, Implement" active-learning format: students were *informed* about the history and importance of a particular problem in context, *experienced* the problem in an active learning lesson, and then *implemented* a program that could help to solve the problem. We feel that this three-part rotation curriculum achieved many of our goals, including instilling in students an excitement about computing and also teaching them the basics of a general-purpose programming language.

IX. Acknowledgements

We would like to thank Google for their generous support for the LEAD summer program. We would also like to thank Carolyn Vallas, Assistant Dean for Diversity, in the School of Engineering and Applied Science at the University of Virginia for facilitating the summer experience.

References

[1] J. M. Cohoon, "Toward improving female retention in the computer science major," *Commun. ACM*, vol. 44, pp. 108–114, May 2001.
[2] *Richard Tapia Celebration of Diversity in Computing*, ACM. ACM, 2003–2011.
[3] J. P. Cohoon, "An introductory course format for promoting diversity and retention," in *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, ser. SIGCSE '07. New York, NY, USA: ACM, 2007, pp. 395–399.
[4] S. Fancsali and L. McGinnis, "Untapped resources: can intellectual diversity promote cultural diversity in technology?" in *Proceedings of the 2005 conference on Diversity in computing*, ser. TAPIA '05. New York, NY, USA: ACM, 2005, pp. 51–52.
[5] A. Woszczynski, C. Beise, M. Myers, and J. Moody, "Diversity and the information technology workforce: an examination of student perceptions," in *Proceedings of the 2003 SIGMIS conference on Computer personnel research: Freedom in Philadelphia–leveraging differences and diversity in the IT workforce*, ser. SIGMIS CPR '03. New York, NY, USA: ACM, 2003, pp. 117–122.
[6] J. M. Cohoon, "Toward improving female retention in the computer science major," *Commun. ACM*, vol. 44, pp. 108–114, May 2001.
[7] L. Carter, "Why students with an apparent aptitude for computer science don't choose to major in computer science," in *SIGCSE Technical Symposium on Computer Science Education*, 2006, pp. 27–31.

[8] C. Wigal, N. Alp, C. McCullough, S. Smullen, and K. Winters, "ACES: introducing girls to and building interest in engineering and computer science careers," in *32nd Annual Frontiers in Education Conference (FIE 2002)*, vol. 2, nov. 2002, pp. F1C–8 – F1C–13.

[9] L. Miller, S. Shearer, and B. Moskal, "Technology camp 101: Stimulating middle school students' interests in computing," in *35th Annual Frontiers in Education Conference (FIE 2005)*, oct. 2005, p. S1F.

[10] J. C. Adams, "Alice, middle schoolers & the imaginary worlds camps," *SIGCSE Bull.*, vol. 39, pp. 307–311, March 2007.

[11] K. R. Cannon, K. A. Panciera, and N. P. Papanikolopoulos, "Second annual robotics summer camp for underrepresented students," *SIGCSE Bull.*, vol. 39, pp. 14–18, June 2007.

[12] P. Doerschuk, J. Liu, and J. Mann, "Pilot summer camps in computing for middle school girls: from organization through assessment," *SIGCSE Bull.*, vol. 39, pp. 4–8, June 2007.

[13] Y. Ouyang and K. Hayden, "A technology infused science summer camp to prepare student leaders in 8th grade classrooms," in *Proceedings of the 41st ACM technical symposium on Computer science education*, ser. SIGCSE '10. New York, NY, USA: ACM, 2010, pp. 229–233.

[14] W. W. Lau, G. Ngai, S. C. Chan, and J. C. Cheung, "Learning programming through fashion and design: a pilot summer course in wearable computing for middle school students," *SIGCSE Bull.*, vol. 41, pp. 504–508, March 2009.

[15] G. Carmichael, "Girls, computer science, and games," *SIGCSE Bull.*, vol. 40, pp. 107–110, November 2008.

[16] U. Wolz, M. Stone, S. M. Pulimood, and K. Pearson, "Computational thinking via interactive journalism in middle school," in *Proceedings of the 41st ACM technical symposium on Computer science education*, ser. SIGCSE '10. New York, NY, USA: ACM, 2010, pp. 239–243.

[17] D. P. Groth, H. H. Hu, B. Lauer, and H. Lee, "Improving computer science diversity through summer camps," *SIGCSE Bull.*, vol. 40, pp. 180–181, March 2008.

[18] I. Pivkina, E. Pontelli, R. Jensen, and J. Haebe, "Young women in computing: lessons learned from an educational & outreach program," *SIGCSE Bull.*, vol. 41, pp. 509–513, March 2009.

[19] S. Graham and C. Latulipe, "Cs girls rock: sparking interest in computer science and debunking the stereotypes," *SIGCSE Bull.*, vol. 35, pp. 322–326, January 2003.

[20] M. B. Rosson, A. Ioujanina, T. Paone, G. Sheasley, H. Sinha, C. Ganoe, J. M. Carroll, and J. Mahar, "A scaffolded introduction to dynamic website development for female high school students," *SIGCSE Bull.*, vol. 41, pp. 226–230, March 2009.

[21] L. Pollock, K. McCoy, S. Carberry, N. Hundigopal, and X. You, "Increasing high school girls' self confidence and awareness of cs through a positive summer experience," *SIGCSE Bull.*, vol. 36, pp. 185–189, March 2004.

[22] J. Gregg, C. McDonnell, and T. Chen, "Work in progress - PEER college summer camp," in *35th Annual Frontiers in Education Conference (FIE 2005)*, oct. 2005, p. S1F.

[23] B. Maxim, W. Grosky, and J. Baugh, "Work in progress - introducing information technology through game design," in *37th Annual Frontiers In Education Conference (FIE 2007)*, oct. 2007, pp. T1B–1 –T1B–2.

[24] A. Denault, J. Kienzle, and J. Vybihal, "Be a computer scientist for a week the mcgill "game programming guru" summer camp," in *38th Annual Frontiers in Education Conference(FIE 2008)*, oct. 2008, pp. T3D–1 –T3D–6.

[25] B. Maxim and B. Elenbogen, "Work in progress - attracting k-12 students to study computing," in *38th Annual Frontiers in Education Conference (FIE 2008)*, oct. 2008, pp. F2H–15 –F2H–16.

[26] M. Al-Bow, D. Austin, J. Edgington, R. Fajardo, J. Fishburn, C. Lara, S. Leutenegger, and S. Meyer, "Using game creation for teaching computer programming to high school students and teachers," *SIGCSE Bull.*, vol. 41, pp. 104–108, July 2009.

[27] S. H. Rodger, J. Hayes, G. Lezin, H. Qin, D. Nelson, R. Tucker, M. Lopez, S. Cooper, W. Dann, and D. Slater, "Engaging middle school teachers and students with alice in a diverse set of subjects," *SIGCSE Bull.*, vol. 41, pp. 271–275, March 2009.

[28] P. A. G. Sivilotti and M. Demirbas, "Introducing middle school girls to fault tolerant computing," in *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, ser. SIGCSE '03. New York, NY, USA: ACM, 2003, pp. 327–331.

[29] J. C. Adams, "Scratching middle schoolers' creative itch," in *Proceedings of the 41st ACM technical symposium on Computer science education*, ser. SIGCSE '10. New York, NY, USA: ACM, 2010, pp. 356–360.

[30] S. Cooper and S. Cunningham, "Teaching computer science in context," *ACM Inroads*, vol. 1, pp. 5–8, March 2010.

[31] M. Guzdial, "Does contextualized computing education help?" *ACM Inroads*, vol. 1, pp. 4–6, December 2010.

[32] L. Layman, L. Williams, and K. Slaten, "Note to self: make assignments meaningful," in *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, ser. SIGCSE '07. New York, NY, USA: ACM, 2007, pp. 459–463.

[33] M. Guzdial and A. E. Tew, "Imagineering inauthentic legitimate peripheral participation: an instructional design approach for motivating computing education," in *Proceedings of the second international workshop on Computing education research*, ser. ICER '06. New York, NY, USA: ACM, 2006, pp. 51–58.

[34] P. Curzon, P. W. McOwan, Q. I. Cutts, and T. Bell, "Enthusing & inspiring with reusable kinaesthetic activities," in *Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education*, ser. ITiCSE '09. New York, NY, USA: ACM, 2009, pp. 94–98.

[35] R. Moreno and R. E. Mayer, "Cognitive principles of multimedia learning: The role of modality and contiguity," *Journal of Educational Psychology*, vol. 91, no. 2, pp. 358–368, June 1999.

[36] F. Coffield, D. Moseley, E. Hall, and K. Ecclestone, *Learning styles and pedagogy in post-16 learning. A systematic and critical review*. London, UK: Learning and Skills Research Centre, 2004.

[37] J. J. McConnell, "Active learning and its use in computer science," in *Proceedings of the 1st conference on Integrating technology into computer science education*, ser. ITiCSE '96. New York, NY, USA: ACM, 1996, pp. 52–54.

[38] J. J. Farrell, R. S. Moog, , and J. N. Spencer, "A guided inquiry general chemistry course," *Journal of Chemistry Education*, vol. 76, pp. 570–574, 1999.

[39] K. D. A. and R. Fry, *Toward an applied theory of experiential learning*. London: John Wiley, 1975.

[40] A. Chickering and Z. Gamson, "Seven principles for good practice in undergraduate education," *AAHE Bulletin*, no. 39, pp. 3–7, 1987.

[41] M. Martyn, "Clickers in the classroom: An active learning approach," *EDUCAUSE Quarterly (EQ)*, vol. 30, no. 2, pp. 71–74, 2007.

[42] E. Douglas and C.-C. Chiu, "Work in progress - use of guided inquiry as an active learning technique in engineering," in *Frontiers in Education Conference, 2009. FIE '09. 39th IEEE*, oct. 2009, pp. 1 –2.

[43] Y. Feaster, L. Segars, S. K. Wahba, and J. O. Hallstrom, "Teaching cs unplugged in the high school (with limited success)," in *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, ser. ITiCSE '11. New York, NY, USA: ACM, 2011, pp. 248–252.

[44] R. Thies and J. Vahrenhold, "Reflections on outreach programs in cs classes: learning objectives for "unplugged" activities," in *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, ser. SIGCSE '12. New York, NY, USA: ACM, 2012, pp. 487–492.