

Making Computer Science Attractive to High School Girls with Computational Thinking Approaches: A Case Study

Oshani Seneviratne

Abstract Computational thinking is a fundamental skill that extends beyond computer science. Conceptually it involves logic, algorithms, patterns, abstraction, and evaluation. The approach for developing a computational mind-set may involve experimenting, creating, debugging, and collaborating. Due to certain implicit biases and societal and cultural factors, girls may not be exposed to these computational thinking concepts and approaches. This has resulted in a decrease in the number of women in computer science since the 1980s. This chapter summarizes some of the challenges faced when teaching introductory computer science to high school girls and the approaches taken to overcome those challenges.

Keywords Gender issues • Computational thinking techniques

Introduction

Gender gap in computer science is a much-studied topic in the recent years (Margolis & Fisher, 2003). According to a report titled “Why So Few?” (Hill, Corbett, & Rose, 2010), only a very small percentage of girls, around 0.4 %, entering college intend to major in computer science, and women only made up 14 % of all computer science graduates, down from 36 % in 1984. In a 2009 poll of young people aged 8–17, only 5 % of girls had said they were interested in an engineering career. Another recent poll found that while 74 % of college-bound boys aged 13–17 said that computer science or computing would be a good college major for them, only a 32 % of their female peers said the same (Association for Computing Machinery; WGBH Educational Foundation, 2009). It has also been shown that, from early adolescence, girls express less interest in math or science careers than boys do (Lapan, Adams, Turner, & Hinkelman, 2000; Turner, 2008). Even girls and women who excel in mathematics often do not want to pursue computer science or any

O. Seneviratne (✉)

Massachusetts Institute of Technology, Cambridge, MA 02139, USA
e-mail: oshani@csail.mit.edu

other STEM fields. Given these disparities, there are many academic programs at various institutions that are trying to address the problem head on and break the glass ceilings in which women may be discouraged in pursuing a career in computer science.

The MIT Women's Technology Program (WTP)¹ is a program that has been running since 2002 with the goal of attracting more high school girls to engineering and computer science. WTP facilitates a rigorous residential summer program for high achieving college bound girls who are either high school juniors or seniors from all over the USA who did not have any prior exposure to computer science. The hallmark of the program is that, it introduces the concepts in a hands-on team-based format with a focus on problem solving. The program has daily lectures, labs with fun team-based projects, and several hours of homework. There are no grades for WTP because the program encourages students to go outside their comfort zones and not worry about a perfect score or making mistakes. During my doctoral studies at MIT, I was very fortunate to teach computer science through WTP to high school girls. The teaching staff included myself who was the main instructor responsible for preparing and delivering all the material, and tutors who help the students during the labs and their homework. The tutors are typically advanced undergraduate students who are majoring in computer science.

Before the summer program began, it was my responsibility, as the instructor, to prepare the curriculum for basics of computer science with comprehensive examples, mini quizzes, and projects the students can try out. Learning in WTP is supposed to be incremental, where the lessons would build up from the previous day. The curriculum covered basic syntax for programming in Python, control structures, functions, object-oriented programming, data structures, algorithms, and recursion. Students were expected to complete challenging conceptual exercises, daily programming assignments, and a final project.

Despite all the preparations we took, when the program started, the teaching staff realized that the students' perception towards computer science needs to be changed to instill a computational thinking mind-set before going ahead with the lessons. The high school girls were previously exposed to concepts such as patterns and abstraction through high school level math and science courses. However, we noticed that some of the students had difficulty in applying such skills they already had to learning basics of computer science theory.

The reasons were twofold: (1) gender-based stereotypes and (2) learning subject matter they were not exposed to before. Although the first issue is only applicable to girls, the second issue is equally applicable to both genders. The following sections illustrate the above-mentioned issues in depth along with the approaches we took to overcome those challenges.

¹<http://wtp.mit.edu/>.

Overcoming the Gender Stereotypes

Overcoming Implicit Gender Bias

As the teaching staff, it was our duty to encourage a supportive community spirit of learning together (Johnson & Johnson, 1987), so that the learning process will not be overwhelming to the students during the fast-paced short summer course. So, in this spirit of learning together, we facilitated a classroom discussion on what computer science meant to the students at the beginning of the course. We asked few open-ended questions like: “Who has met a computer scientist/programmer?”, “What do you think computer scientists do?”, etc. Based on some of the answers, it was clear that some of the students had certain implicit gender stereotypes. A family member, an older friend, or a friend of a friend was “into computers,” but many of them were male, and it seemed as if the students would most certainly equate these people they knew to asocial geeks who keep to themselves typing all day in dark basements and do not see the light of day. As for what they think the computer scientists do, most of the students (rightfully) had the impression that computer science meant coming up with code, but they didn’t equate that to solving problems.

As explained in the “Blindspot” (Banaji & Greenwald, 2013) even among individuals who actively reject gender stereotypes, implicit bias can be common. This bias not only affects individuals’ attitudes toward others but may also influence their own interest in math and science topics. This indirectly hinders a girl’s computational thinking skills. Not only would a girl more likely to associate computer science with men than with women, but she may also encounter negative opinions for women in such “masculine” positions. It was shown that as early as elementary school, children are aware of these stereotypes and can express stereotypical beliefs about which science courses are suitable for females and males (Joyce & Farenga, 2000). Furthermore, girls and young women have been found to be aware of, and negatively affected by, the stereotypical image of a scientist as a man (Luce et al., 2008). Even looking at my own (the author’s) career path as a computer scientist and of my very few female peers, I can see that women face a particular set of difficulties when they are in a male-majority field. The presence of female role models can be hard to come by when you’re one of the only girls in your computer science class.

No Need for Self-Inflicted High Standards

Studies have shown that girls hold themselves to a higher standard in subjects like math (Correll, 2004). Because of this, girls are less likely to believe that they will succeed in a STEM field such as computer science, and therefore, are less likely to express interest in a career in computer science. In a study done in 2005, it was found that gender differences in self-confidence in STEM subjects begin in middle school and increase in high school and college, with girls reporting less confidence

than boys do in their math and science ability (Pajares, 2005). In part, boys develop greater confidence in STEM through experience developing relevant skills, and girls may lose the opportunity to develop such skills due to their vulnerability in losing confidence in STEM areas.

However, WTP was a level playing field, because all the students were girls. Plus, WTP did not emphasize on grades, but rather on learning in a collaborative environment. So, since none of the girls had the pressure to hold themselves to high standards, we encouraged them to be very confident about their abilities, ask questions, and learn from each other.

Female Role Models

As mentioned above, computer science has a bad image among girls, or they were not confident in the skills they already had. Thus, it was clear that we had to address the stereotypes the students associated with computer science. A study reported an increase in girls' interest in computer science and engineering after the girls were exposed to a 20-min narrative delivered by a computer-generated female agent describing the lives of female engineers and the benefits of computer science and engineering careers (Plant, Baylor, Doerr, & Rosenberg-Kima, 2009). Therefore, getting to know female computer scientists who can potentially be the girls' role models can be a huge boost to the girls' self-confidence and increase their interest in the field.

Thus, we decided to get the girls exposed to as many female role models as possible during the program. Unlike in their high school environments, during the summer long program, the students already had lot of access to female role models. The staff members of WTP including the tutors were all female, and the girls felt especially connected to the tutors since they were only few years senior to them. We made sure that the students felt comfortable talking to anybody in the staff during the classroom sessions, during programming labs, and during after-hours in which they completed their homework. Problem solving is an iterative process (Wing, 2006), and acquiring the skills needed to solve or debug a solution to a problem can take a long time or maybe even impossible if the proper support structures are not present, and the students may be discouraged early on and do not develop an interest. This is especially important since the students really need to come out of their comfort zones and experiment with the unknown in order to fix an error in their program. For a novice this can be very intimidating, and thus having access to people who can help them can be very beneficial. The students should not be in a position to give up the entire field if they were not able to correctly write their first program, or debug their code, or do not understand something that can be useful in figuring out the underlying basic CS concepts.

Interacting with women who use computer science in their professional lives gives them an idea of something to go after besides an endless string of code. Therefore, we organized a lunch series throughout the duration of the program and

invited successful female computer scientists, professors, engineers from the industry, and female CS Ph.D. students at MIT, to talk to the high school girls informally about how they first got into computer science and what they are passionate about other than computer science. Most of the speakers were working to solve some very interesting challenges such as finding cures for diseases like cancer, tackling global warming, developing renewable energy sources, developing robots to help the elderly, working on speech synthesis, and understanding the origins of the universe, to name a few. In fact few of the guests did not even identify themselves as computer scientists as their day-to-day work was in some other field such as physics or chemistry. However, they all were influenced by computer science at some point in their lives. Many of the guests had very interesting hobbies, including playing musical instruments, hosting shows on the local radio station, running marathons, and even performing in dance and music festivals! They all had very interesting stories to share about how they got interested in the field and how computer science has helped in their day-to-day lives. After hearing these stories, the girls had several role models to look up to, and most of them later indicated in their WTP exit surveys that this experience positively changed their attitudes toward computer science.

Emphasizing the Importance of Female Presence in Science and Technology Innovation

We also wanted to make it clear to the students that computer science is now a discipline that is playing a key role in invention and creation across all sorts of disciplines from biological science to film and animation. This expansion of the field of computer science and how critical it is across all disciplines increasingly makes it more meaningful to study computer science and related technologies. As computers have become integrated into other disciplines like digital media, including music and film, the geek image has shifted from that of a socially isolated person to include a chic geek image where it can be cool to know about computers. So, the students' perception towards computer science as just "coding" is no longer applicable. Thus, the "geek" image is improving. Movements such as the "#ILookLikeAnEngineer" and "#ILookLikeAProgrammer" hashtag on social media introduce women who contribute to the society in meaningful ways as computer scientists and engineers (Guynn, 2015).

In the classroom, we discussed some examples of the dangers of not having enough female participation in technical roles. For example, some early voice-recognition systems were calibrated to typical male voices. As a result, women's voices were literally unheard. Many of the computer games were designed to cater to young males, and it would be difficult to find games that are equally amenable to both genders. Similar cases are found in many other industries. For instance, a predominantly male group of engineers tailored the first generation of automotive airbags to adult male bodies, resulting in avoidable deaths for women and children

(Margolis & Fisher, 2003). Discussing such imbalances in gender in fields that are near and dear to our lives can be detrimental to our society, and the students seem to understand the broader implications. With a more diverse workforce that includes equal participation from women, scientific and technological products, services, and solutions are likely to be better designed.

Effective Teaching Methodologies

Show and Tell

We wanted to teach the students that programming isn't just about using a particular language. The 1972 Turing award winner Edsger Dijkstra had once said "teaching code to programmers is like teaching how to use telescopes to astronomers" (Haines, 1993). The syntax is vitally important but utterly trivial. Therefore, to help the students understand that computer science is not about typing at the computer all day, or learning some esoteric programming language, we utilized several props in the classroom to convey the message that a computer language is merely a tool. We brought in things like a canvas and a paintbrush to the classroom. Just as these are tools for an artist to paint an imagery that was conceptualized in her mind, the computer is a tool to either express an idea or solve a problem that will be difficult without the tool (i.e., the computer). This kind of "show and tell" approach was very effective throughout the program, as it was very exciting to have physical objects that would not normally belong in a computer science lecture room.

Classroom Discussions

Since many of the concepts in computer science cannot be easily demonstrated using the above-mentioned show and tell approach, we thought of filling in the gap with the day-to-day activities the students engage in using computers. For example, for the very first lesson where algorithms were introduced, we asked the students to get into groups and discuss what things they do in their day-to-day activities that use a computer, what kind of things are easy for a computer to do but hard for a human to do, and vice versa. The students came up with examples such as "web search," "email," and "Facebook." Going by their interest areas, we tried to explain how computer scientists have made those services work. Search engines such as Google use algorithms to put a set of search results into order, so that more often than not the result we're looking for is at the top of the front page. Likewise, the Facebook news feed is derived from our friends' status updates and other activity, but it only shows that activity which the algorithm thinks we will be most interested in seeing. The recommendations we get from Amazon, Netflix, and eBay are algorithmically

generated, based in part on what other people are interested in. Given the extent to which so much of our lives are affected by algorithms, we iterated the importance of learning algorithms, so that they can also create a novel algorithm that can help solve a problem.

Based on some of the initial answers we got, we realized that the students thought computers are only those devices that have screens, keyboards, and mice in addition to the microprocessor, memory, etc. They did not know other household devices that they had access to, such as calculators, smartphones, cars, and Roombas as “computers”. Therefore, we wanted to illustrate the ubiquity of computer science. The calculator app on our smartphones is able to perform complex calculations that would take a normal human a considerable amount of time to compute, the GPS in our cars is able to tell us directions, and there are other such examples where the computing capabilities that are already readily available to the students are easily overlooked: Computers in our cars help us with cruise control and to display information based on the inputs to its sensors; Roombas in our houses clean the floor in an autonomous manner without any human intervention whatsoever; gaming consoles are able to load the programs and respond to the inputs from the joysticks or even use our body movements in the case of innovations like Kinect.

Discussion on these everyday-computing devices proved to be a very good exercise and a good entry point to explain what computer programs are capable of. All such devices that have a computer inside need to be programmed using an algorithm. Even a simple application such as the calculator needs the user to understand and interpret the problem before the calculator can help out with the arithmetic.

Examples First, Theory Later

Some research studies have found that men outscore women by a medium to large margin in the area of spatial skills, specifically on measures of mental rotation (Linn & Petersen, 1985) (Voyer, Voyer, & Bryden, 1995). Well-developed three-dimensional spatial-visualization skills are a must for subfields of computer science such as robotics and computer graphics. However, studies have found that spatial skills are not innate but developed (Sorby & Baartmans, 2000). Lego Mindstorms where students can take things apart and put them back together again and do visual block programming can greatly help develop these essential spatial skills. In our experience, many computer science programs often focus on technical aspects of programming early in the curriculum with a strong focus on theory and without much focus on the applications of the concepts. This can be a deterrent to students, who may be interested in broader, multidisciplinary applications. Thus, during the course of WTP, we always made it a point to talk about the applications; no matter how trivial they might be related to the topic, the students are learning.

Teaching Algorithms

When delivering the lessons beyond these introductory concepts, we always made it a point to start the lecture with a fun activity related to the lesson. For example, to illustrate what an algorithm is like and how they can get started to conceptualize algorithms, we asked a volunteer to explain how to write a program to make a peanut butter and jelly sandwich. We brought the ingredients necessary to make the sandwich to the class; and a staff member was acting as the computer, and the volunteer was the programmer. The student volunteer had to give the staff member the exact steps as to how to make the sandwich. The end goal was the delicious sandwich.

The instructions have to be “programmed” in a certain order, and if it is not in the desired order, the computer (i.e., the staff member) will not perform the actions. For example, if the student said, “spread jelly on bread,” and at that time the jelly jar was not open, the staff member will not perform any activity since the jelly was not reachable. Instead, the staff member will make a funny face to indicate the error. Once the student realized the mistake and mentioned “open jelly jar” before “spread jelly on bread,” the staff member would perform the activities, and the volunteer got the peanut butter and jelly sandwich as reward. Even though this was a very simple example, students enjoyed this exercise very much, and they really got the idea that writing a program is like writing a recipe and can be very relatable to the activities we perform in our day-to-day lives.

Teaching Loops and Conditionals

Another such activity involved marching as a preamble to loops and conditionals, where a student had to follow a path through the classroom to reach a certain destination based on the instructions given. Then the class was asked to come up with the pseudocode to illustrate what the student volunteer performed. This resulted in a very engaging atmosphere, where students were able to conceptualize the program, and also were able to discuss their code in a collective manner.

Teaching Functions

We had the most interesting set of challenges when teaching functions. Some of the popular mistakes that students made were confusing the “return” statement with “print” (in python), not understanding that a function remembers where it came from and goes back there when it is done, and that a function can be used to encapsulate blocks of code. Many of these concepts were a bit abstract for many of the students. With an example involving dessert recipes, we were able to illustrate that; for example, a chef can delegate different parts of a dessert (say, tiramisu()) to other worker chefs. These worker chefs can act as routines that create subparts of the

dessert like ladyfingers() and cream(). If the chef wants to make another dessert such as ice cream() or parfait(), she can call the same cream() function that was used in tiramisu(). Although it was tempting to introduce things like stacks when teaching functions, we thought that it may be a bit too overwhelming for these beginner level students at WTP. It should be introduced only at an advanced level or if a student questions about the inner workings of function calls.

Teaching Sorting

For the lesson on sorting, going with the same approach of a fun activity before class, we asked several students to line up and asked the rest of the class to come up with a way to order them according to their height. To our surprise, the students came up with bubble and selection sort algorithms by themselves! We also utilized online videos available on different sort techniques before diving into the nuts and bolts of implementing the algorithms. Visuals are powerful tools in the classroom, and there is no better way to teach such abstract concepts.

Live Examples

In order to complete the homework assignments, the students had to use Linux machines. But many of them were not familiar with the operating system. Therefore, the teaching staff decided to do a live demonstration of the system. The staff member would perform something, and the students were expected to follow. We also did a live debugging session during class, highlighting the instructor's thought process that went in to fixing the problem and also showing how to use the tools that are at the disposal to them.

Let's Build a Game!

While many parents often worry about recreational "screen time", some educators now believe that gaming could be a way to get girls interested in coding and even to increase the numbers of girls in computer science. Therefore, we decided to have the students implement a Tetris game for the final project. Even though the task of implementing a game seems daunting, most of the components that were already required were completed in previous lab sessions. However, since the project may seem overly ambitious especially given all the difficulties some of the students were having during the previous lab sessions, we told them to work in pairs for the project. All the groups had a working game in the end and had time to play with it in class. Pretty much all of them were excited to have the working product in the end.

Results

Through several classroom activities, we managed to teach some of the core principles of computational thinking that many of the girls already knew by intuition: Logic is essential for predicting and analyzing things we want to be computed, algorithms to make steps and rules about executing an idea, decomposition to breaking down a problem to manageable chunks, patterns for spotting and using similarities, abstraction for removing unnecessary details from a given problem and generalizing to fit a broader class of problems, and, finally, evaluation to verify whether the solution worked for a given problem or not (Wing, 2006).

Since the start of MIT WTP in 2002, over 500 students have participated in the program so far. According to the WTP Director and WTP-EECS Track Coordinator, Cynthia Skier, of the 431 students who have participated in WTP over the years, over 64 % are in a field of engineering or computer science, while another 21 % are in math or science fields. Furthermore, the numbers from the 2015 cohort show increased enthusiasm in computer science toward the end of the program. The students were asked about their perception toward computer science and STEM in general, both before and after the commencement of WTP. The percentage of students definitely planning to take college classes in computer science moved from 38 % before starting WTP to 80 % after completing WTP. 60 % listed CS as a probable college major in the exit surveys, which was a 50 % increase over the numbers in the entrance surveys. Some of the answers to qualitative questions in the exit survey indicated that the students got a better understanding and a better outlook on CS after the program. Many students liked the lunch series where we brought in female role models to talk to the students, the activities conducted before the lessons, and the final project where the students had to build a functioning computer game.

This upward trend in girls' interest in computer science is evident at the national level as well. In 2013 girls only made up 18.5 % of A.P. computer science test takers nationwide². In three states, no girls took the test at all. During the recent years, these numbers have been growing steadily. In 2014 25 % of the A.P. computer science test takers were female³, and in 2015 that number increased to 27%⁴. While these numbers are not representative of the population, many efforts by educators and nonprofit organizations seem to have made positive impact in making computer science attractive to girls. Over the years, WTP has produced many shining stars who were equipped with the necessary computational thinking skills. One of the best examples is Tamara Broderick, who in 2002 completed WTP as a high school student and in 2015 returned to MIT as an assistant professor⁵.

²<http://research.collegeboard.org/programs/ap/data/participation/ap-2015>.

³<http://research.collegeboard.org/programs/ap/data/archived/ap-2014>.

⁴<http://research.collegeboard.org/programs/ap/data/participation/ap-2015>.

⁵<https://www.eecs.mit.edu/news-events/media/tamara-broderick-woman-technology>.

Conclusion

Utilizing computational thinking approaches coupled with strong role models can especially be useful for getting young girls interested in computer science. It is our belief that girls are easily discouraged from computer science due to many reasons that are not related to their personal capabilities. Even though the girls have many computational thinking skills, they are either not aware of them or hold themselves to very high standards.

From our experience with WTP, the approaches to instill computational thinking in high school girls who have not been exposed to computer science previously are numerous: First, the students should be guided in an encouraging manner with mentorship, fun activities, and ample computer science related exercises. Second, they should be encouraged to tinker with a solution by iterating and playing around with multiple solutions ready to throw away any solution if needed. Third, the debugging process should be mastered, where the focus is to find and fix the errors introduced from the creation and tinkering processes, without getting frustrated or losing sight of the end goal. Thus, a collaborative and supportive environment with plenty of guidance can help gain the necessary skills to think like a computer scientist and achieve one's full potential.

References

- Association for Computing Machinery; WGBH Educational Foundation (2009). *New Image for Computing Report*.
- Banaji, M. R., & Greenwald, A. G. (2013). *Blindspot: Hidden biases of good people*. New York: Delacorte Press.
- Correll, S. J. (2004). Constraints into preferences: Gender, status, and emerging career aspirations. *American Sociological Review*, 69, 93.
- Guynn, J. (2015). *#ILookLikeAnEngineer challenges stereotypes*. *USA TODAY* Published: August 4, 2015, <http://www.usatoday.com/story/tech/2015/08/03/isis-wenger-tech-sexism-stereotypes-ilooklikeanengineer/31088413/>.
- Haines, M. D. (1993). Distributed runtime support for task and data management. Ph.D. Dissertation, Colorado State University.
- Hill, C., Corbett, C., & Rose, A. S. (2010). *Why so few? Women in science, technology, engineering, and mathematics*. Washington, DC: AAUW.
- Johnson, D. W., & Johnson, R. T. (1987). *Learning together and alone: Cooperative, competitive, and individualistic learning*. Englewood Cliffs, NJ: Prentice-Hall.
- Joyce, B. A., & Farenga, S. J. (2000). Young girls in science: Academic ability, perceptions and future participation in science. *Roeper Review*, 22, 261.
- Lapan, R. T., Adams, A., Turner, S., & Hinkelman, J. M. (2000). Seventh graders' vocational interest and efficacy expectation patterns. *Journal of Career Development*, 26, 215.
- Linn, M. C., & Petersen, A. C. (1985). Emergence and characterization of sex differences in spatial ability: A meta-analysis. *Child Development*, 56, 1479.
- Luce, S. A., Servon, C., Sherbin, L. J., Shiller, L., Sosnovich, P., & Sumberg, E. (2008). The athena factor: Reversing the brain drain in science, engineering and technology. *Harvard Business Review Research Report*.

- Margolis, J., & Fisher, A. (2003). *Unlocking the clubhouse*. Cambridge, MA: MIT.
- Pajares, F. (2005). Gender differences in mathematics self-efficacy beliefs. In *Gender differences in mathematics: An integrative psychological approach*. New York: Cambridge University Press.
- Plant, E. A., Baylor, A. L., Doerr, C. E., & Rosenberg-Kima, R. B. (2009). Changing middleschool students' attitudes and performance regarding engineering with computerbased social models. *Computers and Education*, 53, 209.
- Sorby, S. A., & Baartmans, B. J. (2000). The development and assessment of a course for enhancing the 3-D spatial visualization skills of first year engineering students. *Journal of Engineering Education*, 89, 301.
- Turner, S. L. (2008). Gender differences in Holland vocational personality types: Implications for school counselors. *Professional School Counseling*, 11, 317.
- Voyer, D., Voyer, S., & Bryden, M. P. (1995). Magnitude of sex differences in spatial abilities: A meta-analysis and consideration of critical variables. *Psychological Bulletin*, 117, 250.
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49, 33–35.

Emerging Research, Practice, and Policy on
Computational Thinking

Rich, P.; Hodges, C.B. (Eds.)

2017, XXVII, 413 p. 60 illus., 41 illus. in color., Hardcover

ISBN: 978-3-319-52690-4