I was drawn to teaching from a young age, from helping my friends with their homework as a grade schooler, to being a paid tutor, to teaching classes as an undergrad and a grad student, to developing curricula for special summer sessions. Some of my techniques have worked, and others haven't. This statement presents some of my central ideas related to teaching, and the attraction and retention of qualified students.

As part of my education, I have completed a Delta Teaching Certificate at the University of Wisconsin. The Delta program (`http://www.delta.wisc.edu/`) is a curriculum for science and math teachers at the university. It emphasizes three pillars: teaching-as-research, learning-through-diversity, and creating a learning community. As part of the program I have taken classes and completed an internship which focused on improving my teaching skills, in addition to those classes required for my PhD.

Pay Individual Attention to Students

I believe that the best way to create an inclusive environment is to learn every individual student's needs, and to be as available as possible to him or her. Rightly or wrongly, computer science often suffers the reputation as being an "old boys network" into which it is difficult to gain admission. By being available, this impression can be minimized.

The first step toward doing this is to learn every student's name. Assuming the class is small enough, the teacher needs to make an effort to ask every student a question during every lecture—especially those students hiding in the back. If the class is too large to call on every student every time, the teacher still needs to make a best effort. This can be coupled with an open-door policy and responding to students' needs as promptly as possible. This does more than help the students to understand difficult concepts from class. It shows them that somebody cares if they pass or fail, and that there is an academic community that welcomes them.

One also needs to realize that students' schedules are often very different than those of their teachers. When I am teaching, I make it a point to check e-mail after midnight on every working day, and often on weekends as well. I have found that students appreciate these prompt responses when most of the world has gone to bed.

My first semester as a TA, one of my students was having trouble deciding on a major. He thought that his previous schooling didn't prepare him well enough for computer science. I listened to him and answered his late night e-mails. At the end of the semester he sent me a note thanking me for my help, and said that I was one of the best TAs that he'd ever had. In a lecture class of over a hundred students, I had helped him to feel more grounded and more that he mattered. I think that this is one of the best ways to retain intelligent students who might otherwise leave the field.

Modular, Interactive Lessons

One of the most promising changes in education over the last few years has been the development of interactive applets that can be used in the classroom. Such small programs are ideal for illustrating many of the algorithms and techniques used in computer science. Sorting algorithms, hash tables, and binary trees are all concepts that can be much more clearly shown with the use of these interactive illustrations. Further, because they are modular, they can be dropped into lectures with very little work. And because other teachers are designing them too, there is a large library already in existence.

I designed one such module as a part of my Delta Teaching Certificate. It is an interactive virtual planetarium called cSky that I created using Adobe's Flash environment. The program shows how the night sky moves over time from any location on the earth. Using it, a teacher can demonstrate the cause of the seasons, the sun's path through the ecliptic plane, the wanderings of the planets, and many other basic astronomical concepts. It can be dropped seamlessly into the middle of a lecture for use during class, or students can visit a web page with it embedded and try it out for themselves. To test it, I conducted a trial of college non-major astronomy students. Self-reported answers indicated that not only did it significantly improve students' understanding of the material, but it also erased a gender-gap that had been present in

the pre-testing. cSky can be seen online at `http://pages.cs.wisc.edu/~aasmith/csky/csky.swf`.

Concrete, Problem-Based Lessons

Computer science can easily appear abstract and arcane to students. This can be especially true in the first year, when they have little idea of why computers don't just "work the first time." I feel that, for such students, it is important to ground lessons in some concrete problem. Telling students to sort an array of numbers or create a certain data structure are lessons that will be easily forgotten. Giving them an open-ended scenario helps them to think more about why programmers make the decisions they do. Even more important, they encourage the student to elaborate on the program, taking it in directions that the teacher hadn't forseen.

Small games are ideal for this. Not only do they ground the problems within a simple framework, but they give the student a chance to have some fun. If a lesson is enjoyable, it will be remembered—perhaps for years. I first used this technique by using the children's game Hangman in order to teach about string manipulation. However the technique can be taken much farther. For example, a student learning data structures can make a filter for a word game that only recognizes valid English words, using binary search or suffix trees. Students in an Artificial Intelligence class can each design an agent for a simple yet open-ended game like Mancala or Reversi ("Othello"), and even compete against each other.

The idea in each of these cases is not only to give the student a better understanding of the programming process, but to make him or her more personally invested in the final product. Pride in a task becomes joy, and the joy will turn to devotion.

The Right Tool for the Right Job

I believe in using technology in the classroom, but I believe its use needs to be thought out carefully. For example, many teachers have switched from using a chalkboard to a slideshow program such as Microsoft's PowerPoint. This has provided several advantages: lectures become more easily available, teachers can quickly modify slides as times change, and the material is more easily shown to students in a large lecture hall. However it has also created some problems. The programs were made for a business environment and not an academic one. Slides often go by too rapidly. Students end up relying on complete printouts of the slides and not on their own notes. Sometimes they don't even attend class, because they can just download the notes. Further, diagrams are often awkward, and equations can be difficult to include.

My solution to this problem is to teach using HTML web pages and a standard web browser. I have spent a considerable amount of time developing tools to do just that. This offers several advantages over the current methods. For example, one can scroll smoothly through the lecture rather than having to jump between discrete slides, giving the students more time to take down their notes. HTML is also easily incorporated with MathML or LaTeX, two technologies that allow equations much more easily than Microsoft's equation editor. Interactive applets (such as my cSky program) can be quickly dropped into a lecture with little effort. And because HTML is a well-known standard, lectures prepared this way could easily be readable for years longer than slides created in a proprietary format.

This example illustrates my philosophy toward using technology in the classroom. There are many advantages to its use, but it should not be used blindly. Rather, a well-balanced approach can help create a learning experience that incorporates the best features of many different teaching methods.

Continued Refinement of Teaching Methods

Underlying all of this is a spirit of continued contemplation, and of combining the old tried-and-true methods with the newer ones that show promise. None of these ideas came quickly. They came about through years of thought, continued discussions with other teachers, and observing countless lectures by others. In short, it is a process that does not end. I fully expect my ideas to undergo some transformation over the years to

come, as I teach more. However I believe flexibility to be a virtue. I will continue to refine my methods, retiring some and further expounding upon others. With luck, I hope to still be asking "What makes a good teacher?" many decades down the road.