CS-480—Senior Seminar
Synopsis: *Current methods in teaching Computer Science*
Fall '03

Micah Acinapura, Julie Picket

10/Sept.

Primary Papers

Guzdial, Mark and Soloway, Elliot "Teaching the Nintendo Generation to Program. " *Communications of the ACM*, Vol. 45, No. 4, pp. 17–21, April 2002.
van Valkenburg, Mac E "An Engineering Curriculum for the Future." *IEEE Communications Magazine*, (Volume and number missing), pp. 12–15, Dec 1990.
Walker, Henry M. and Schneider, G. Michael "A Revised Model Curriculum for a Liberal Arts Degree in Computer Science." *Communications of the ACM*, Vol. 39, No. 12, pp. 85–95, Dec 1996.

Additional Papers

Katz, Kaila "Historical Content in Computer Science Texts: A Concern." *IEEE Annals of the History of Computing*, Vol. 19, No. 1, pp. 16–19, 1997.

Short Title of Paper or Author's Name

Synopsis

Our two main articles discussed issues with current CS curricula. While Guzdial and Soloway was not particularly in depth, it still brought up issues that Walker and Schneider and van Valkenburg addressed as well. The main issue dealt with what curricula (specifically CS and engineering) should do to introduce students to the subject. Guzdial and Soloway and van Valkenburg were aiming for the idea that in order to engage students in a subject, a curriculum must start with classes that students would consider fun or interesting as opposed to classes that would teach basic skills and prepare student for more sophisticated upper level courses. Guzdial and Soloway add that the standard 'Hello World' introduction to programming just does not impress students anymore. They indicated that it was a change in the media/medium of courses that was necessary, as opposed to van Valkenburg who felt that the underlying structure of courses and curricula that needed modification. Walker and Schneider had a much more in depth look at CS curricula, especially in colleges like ours. They reviewed not only the content of classes but also time commitments for students, faculty needs, as well as the need for students to study other subjects. Walker and Schneider also asset the entire structure of a liberal arts CS curriculum from the intro courses to core subjects and electives. Their suggestions were revised editions of past curricula, with valid points. Walker and Schneider differed in saying that before students can get a good grasp of complex higher level classes they should have a good base knowledge, obtained from core skills courses.

Our discussion covered mostly topics brought up by Guzdial and Soloway, but grazed on Walker and Schneider as well. One issue that seemed to be at the heart of the problem was the complexity of modern computing. Current desktop computers and computer programs are very powerful and full of vibrant graphics and media. Where as most CS1 students get to use graphics in such programs as text-based hangman. They are producing very small programs on machines that are capable of exponentially more than they are able to accomplish. This can be very deterring to a beginning student. Students also may not be impressed by 'Hello World' because they use much more sophisticated software everyday for trivial tasks. We decided that a good approach would to be to introduce students with programs that expressed the same ideas as 'Hello world' but with more substantially modern interests at heart. We were also interested in not only interesting typical beginning computer scientists, but more creative 'bricoleurs' as well. A good example was the pre-CS1 class at Kalamazoo where the students first manipulate classes of objects to create a fish tank. Within our example, we brainstormed several ways to do this, namely, to allow for a very large number of variables at the command-line level, and allow these students to simply explore the meanings and values of these variables which could do things such as alter the color or size of the fish, to change the direction or path of the fish, and to adjust the initial placement of the fish. Though it is agreed that these ideas may be more involving for students in a beginning class, at the end of the day we were left with some outstanding questions;

Should we teach this 'Nintendo Generation' to program? If these students are not interested in the fundamental concepts of computer science, are they going to be reliable additions to the field?

How much can we change the media of a class without changing the content, or what is actually taught in the class?

Are we capable of fully answering these questions as a group? Most of us grew up in said 'Nintendo Generation' and have a hard time balancing the more interesting ideas with the concept of grasping fundamental issues in the curriculum.