Introductory Computing Across Engineering Disciplines

Mark Sherriff

University of Virginia / Department of Computer Science

Introduction and Objectives

Introductory Computer Science courses can be a challenge for many engineering schools. In many schools, CS1 is a required course in the engineering core. However, the stakeholders in the course (CS faculty, engineering faculty, and students) often have potentially opposing views on what the course should focus on.

Should the course focus on:

- Fundamental concepts in programming?
- Tools needed in various engineering disciplines?
- Computational modeling?

How can one course serve as both the introduction to four different majors (Computer Science, Computer Engineering, Electrical Engineering, and Systems Engineering) and yet still fulfill all the tools and technical needs of the rest of the engineering school, all while keeping the students engaged?

Developmental History of Innovation

At UVa, our CS1 course has pressure from many sides to change in some way. For those students interested in a "computing-related" major, taking a CS1 course during their first year makes a great deal of sense. They get some fundamentals of programming and problem solving in Java early and will be prepared to take the second required course in CS the next semester (all of these majors require at least two courses in CS). The faculty in the CS department in general also see this as a boon, as we get to expose nearly every incoming student to the principles of computing early on in their career, which we believe can be beneficial regardless of the major they end up choosing.

However, there is pressure for change from the other departments in the School of Engineering and Applied Science. Many faculty in these departments want to see the introductory CS course to be much more of a tools course, specifically in MATLAB. Faculty in other departments are concerned that when students get to their domain-specific computing and modeling courses, the students cannot remember how simple programming constructs work. However, these students usually haven't done any programming in almost two years by the time they get to these later courses, so it isn't surprising that they are rusty on these concepts, but it is viewed more as a deficiency of the CS curriculum. Thus the other majors create their own "programming add-on" courses that their majors take in addition to the standard CS1 course. This creates effectively a duplication of effort and some level of resentment between departments.

So the question remains: how can we meet all stakeholder needs?

Learning Activities and Materials / Execution

To try to address these concerns, our CS1 course currently is taught in a problem-based, active-learning environment. Students use pair programming in labs and work in groups to complete relatively complex programs that have targeted learning objectives that are relevant to other disciplines. Below are screenshots from two sample assignments.

One deals with reading a large (>50000 row) data set - a task found in many engineering disciplines. In this assignment, students plot a travel route using Google's static map API and then search a list of points of interest to find out what places they should stop along their route.

The second assignment deals with pathfinding, artificial intelligence, and object orientation, concepts common in upper-level CS courses. In this zombie apocalypse game, the player must avoid the zombies for as long as possible. Students must program both player control and zombie AI along with all the game constructs.

Engaging assignments such as these help drive the student, while the underlying concepts are still related to computer science and engineering at large.

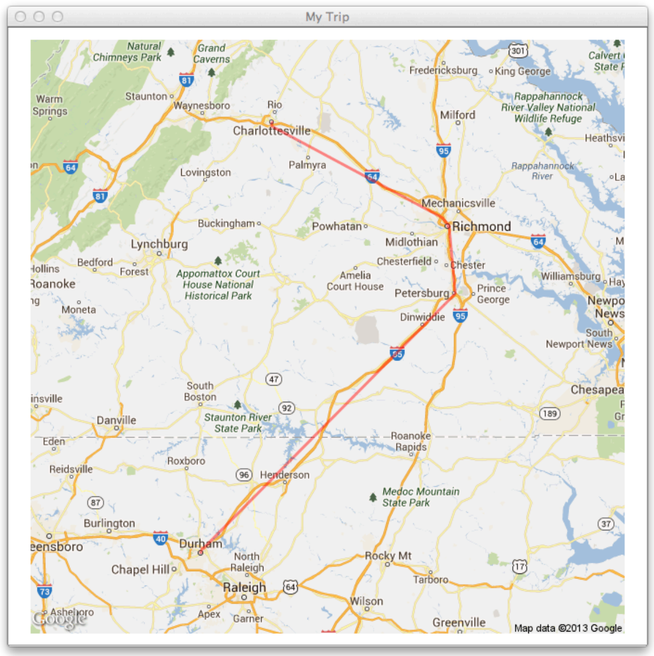

Figure 1. Example Google Map Static API route assignment

Discussion

Our students have shown us that they enjoy and respond well to this type of teaching methodology. Students are successful in later CS courses, indicating that they did learn the material necessary to move on to later courses.

Anecdotal evidence from other faculty still indicates that some students that enter other majors do not perform adequately in computing-driven courses. Whether this is a tools issue (Java vs. MATLAB), a learning outcome issue, or just the perception of the faculty is not quite clear.

Our goal at UVa is to continue to revamp the introductory core engineering experience, including computer science. One idea that is being considered is a combination CS-Applied Mathematics two-semester course on applied computing and data analysis.

Major Issues to Resolve

The main question for the attendees at FOEE is this:

What role should computer science play in the core engineering curriculum?

Should computing just be taught in discipline-specific environments? Is it unreasonable to use an intro course for the purposes of the major? Where should the line be drawn between "tools course" and "computing course"? Can these concepts co-exist effectively?

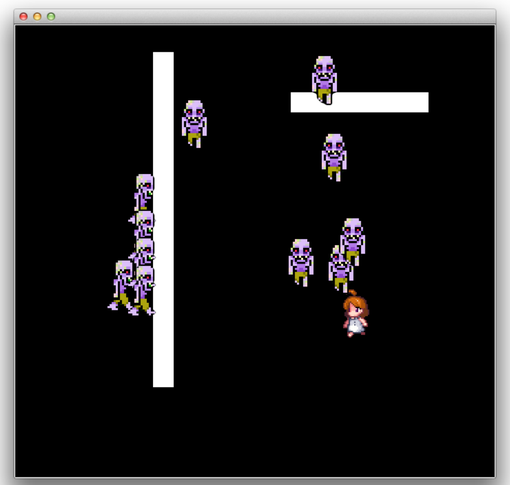

Figure 2. Example zombie apocalypse assignment

Acknowledgments

Computer Science at the University of Virginia

NATIONAL ACADEMY OF ENGINEERING OF THE NATIONAL ACADEMIES

2013 Frontiers of Engineering Education

Irvine, California

October 27-30

Sponsored by:

The National Academy of Engineering and

John McDonnell and the McDonnell Family Foundation