Ravi Chugh | Teaching Statement

Through classes, advising, and curriculum development, I aim to teach fundamental computational thinking skills that will serve students with diverse interests as they pursue careers in technology and other sectors.

Teaching Philosophy

Besides choosing concepts that I think are important for students to see in the classroom, I view my job as teaching students how to think about these ideas. I attempt to do this by structuring most of my classes as live coding sessions, where I lead students through programming exercises to motivate and introduce new ideas, soliciting class participation continuously along the way. Given that my undergraduate classes have had been between 25 and 50 students, I am able to point out connections among comments and questions that specific students have made in previous meetings. By encouraging students to participate and guide the discussion, I hope to provide some basis for students to master the process of unraveling a problem and working towards a finished solution. There are pitfalls with the live coding approach: occasionally a syntax error or compiler flag exposes a corner of a language that I am unfamiliar with, or a student-driven solution takes us down an unexpected path—each of which may result in a longer overall route to the solution that I intended. Nevertheless, I believe the occasional pothole is a price worth paying to have students help navigate most discussions.

> *"I am a graduate student and I have taken numerous classes in my ... career. Ravi is, hands down, one of the best instructors I have ever had! His teaching philosophy of writing the code as he lectures is simply amazing, because he really engages students in the development of the algorithms, data structures, functions, etc. He has a unique charisma of being able to incite interest and curiosity even for the most perplexing of topics (e.g. monads and monad transformers)... He is, simply put, inspiring!"*
>
> — Anonymous Student
> Course Evaluations, Honors Introduction to Computer Science I (Fall 2016)

This classroom style works well, I think, in classes with relatively few students and where the topics lend themselves to programming exercises that are small enough that the entire scope can be adequately covered within the time constraints of class. As I continue to teach classes with different topics and larger numbers of students, I plan to experiment with different lesson structures. For example, as the instructor for an undergraduate course at University of California, San Diego with approximately 100 students, I incorporated the use of clickers during each lecture to help keep the large number of students, with different backgrounds and expertise, engaged during class. Looking forward, I am interested in experimenting with techniques such as pre-recording lectures, flipping the classroom, and creating additional forms of lecture notes and other material to accompany interactive class meetings. We are witnessing a time when more and more resources are openly available to large groups of students; I hope to contribute valuable course material, as well as innovations that help others create such course material, throughout my teaching career.

Research Mentoring Philosophy

My research interests have provided opportunities for a variety of program analysis and user interface design and engineering projects. So far, in my first three years, I have worked closely with one Ph.D. student and eight undergraduate students on research. My Ph.D. and undergraduate students have co-authored three papers, two of which are published in top PL and HCI venues and one which is in preparation. I meet regularly and often with my students in order to set concrete, short-term goals that are realistic and to monitor progress and provide guidance. I believe this is especially important early on, because it takes practice to identify which parts of a problem are crucial and which parts can lead to wasted effort and frustration. Helping a student focus on the "essence" of a problem can lead to more rapid progress, giving the student the confidence to continue and discover his or her own particular interests and strengths. I also stay closely involved with the programming implementations of most student projects, both because I enjoy it and because it helps me to organize a variety of interconnected projects, which often progress at different rates due to all the competing demands on everyone's time.

Curriculum Development

The goal of my research—to combine programming languages and direct manipulation user interfaces—opens the possibility for new courses, both for students in computer science as well as in other disciplines. Within the Computer Science Department, there are a couple of courses—including a Functional Programming course I designed as well as a Data Visualization course taught by another professor—that involve web programming assignments using technologies of the day, such as, JavaScript, D3, and Elm. In the future, I plan to retarget some of this course material and assignments to use the direct manipulation programming systems, such as Sketch-n-Sketch, developed in my research group. This experience will help evaluate our research results among expert programmers, and has the potential to inspire students to seek out, or develop, similar technologies in their chosen careers.

Outside of the Computer Science Department, I plan to use familiar application domains to motivate and teach programming. In graphic design, students often use tools like Adobe Illustrator. In industry, many designers also learn to program in Processing or p5.js to create some work programmatically. I plan to design a course that uses the integrated combination of programming and direct manipulation in Sketch-n-Sketch to show how these techniques complement and augment each other. This may preview the kinds of technology that designers and artists may encounter in the future. To integrate such topics effectively, I plan to work with instructors in other disciplines to understand the needs and goals of students whose primary objective is not learning how to design programs. As computational thinking continues to pervade our society, I hope that direct manipulation programming systems prove useful in making it more intuitive and accessible to teach programming to a wide variety of students.