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 disciplines.

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. 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.

For course material and class sizes that do not readily fit this mold, I plan to experiment with different lesson structures. For example, when teaching a course with approximately 100 students at University of California, San Diego, I incorporated the use of clickers during lectures to help keep students engaged. Looking forward, I will continue to experiment 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. These types of activities have rapidly become more important given the pandemic crisis, and the changes we are all currently working through may have lasting effects on standards and best practices for years to come. As 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 Advising

My research program has provided opportunities for several programming language and user interface research, design, and software development projects. In six years at the University of Chicago, I have been fortunate to work closely with three Ph.D. students, ten undergraduate students, and two postdoctoral researchers. These collaborations have led to co-authored publications, opportunities to attend and present at academic conferences, and professional training for subsequent careers in research or industry.

Throughout my own career, I have benefited from the mentorship of many supportive teachers and colleagues, and from social, educational, and economic opportunities that are not available to all. To help pay these privileges forward, I will aim to advise and mentor students representing a broad range of backgrounds and experiences, and to support their personal and career growth.

Curriculum Development with “Computational Canvases”

My research aims to build “bidirectional” connections between code and direct manipulation GUIs, thus creating “computational canvases” for making a variety of digital objects. As software continues to be integrated into professional and personal spheres, I hope that these computational canvases serve as powerful tools for teaching computational thinking to a wide variety of students.

“Creative Coding.” Starting in Spring 2021, I will teach a new introductory programming course for college students interested in media arts, design, and hopefully many other subjects. Traditionally, graphic designers primarily work in systems like Adobe Illustrator; some also learn to write code in languages, such as Processing and p5.js, to programmatically generate some designs. In Creative Coding, we will use tools that—like Sketch-n-Sketch—blend programming and direct manipulation throughout the curriculum. Students will learn how these usually disparate systems complement and augment each other, and they will be empowered to view software as extensible, programmable platforms rather than opaque “apps” built and bundled only by expert programmers. In addition to this course for college students, I will explore how the material and tools can be translated effectively for different student groups, such as middle school and high school students, as well as remote learners.

"Everyday Coding." Similarly, I am interested in developing a new course where office applications—such as word processors, spreadsheets, calendars, and email clients—are augmented with support for interactive programming and used to teach how programming can help tackle tasks that are inefficient or difficult—if not impossible—using existing GUI-based software.

Professional Software Development. New interactive programming tools can, of course, also be incorporated into courses for computer science students in particular; obvious candidate topics include data science, data visualization, and web development. Incorporating computational canvases into classrooms may inform and inspire those University of Chicago students who go on to design future software technologies.