John DeNero

Teaching Statement

I love teaching, both because of the challenge of creating excitement about a subject, and because I take great pride in the impact I make on my students. At Berkeley, I have focused on helping to create a great undergraduate artificial intelligence course, known to locals as CS 188. Creating a course is like building a house: it requires a sound plan, quality tools, and skilled execution. I have encountered many interesting challenges related to each of these ingredients during four years of teaching this course. I have also learned that a well-designed course becomes a great course in the same way that a house becomes a home: by that twist of personality and creativity that makes a course (or house) unique. In every role that I have filled while developing CS 188, I have sought to add character to the course and deliver an excellent learning experience to students. In turn, each role has taught me critical lessons about the process of learning that have contributed to the dedicated, enthusiastic instructor I am today.

I began teaching as the instructional equivalent of a carpenter: leading discussion sections that reinforced and repaired students' understanding of lecture. I wanted students to engage, so I invented zany problems for students to solve in small groups. In my classes, search algorithms were about flipping pancakes, and propositional logic encoded the students' seating assignments. My students enjoyed this approach: I won the outstanding graduate student instructor award in computer science, and a similar award from the university, both based on student feedback. An email I received after my first semester read, "*I really appreciated all the extra help you gave us in CS 188 in the Fall. You pretty much taught the class. I'm incredibly thankful you were a TA.*"

Clever problems are fun, of course, but they also provided pedagogical benefits beyond mere entertainment. First, creative problems demonstrate the breadth of applicability of computer science techniques. A student's tool chest of algorithms and data structures is only as useful as his or her ability to recognize where and how each tool can be applied. Second, working through problems is the ultimate diagnostic tool for ferreting out misunderstandings. Students often know when they are confused, but rarely why. Instructors need to figure out what students currently believe, how it differs from what they should believe, and what patch—in the form of a clarification, example problem, or illustration—will repair the confusion. I found that providing this service to students made a substantial impact on their understanding.

I wanted more of this positive impact, but I could only hold so many extra office hours. (I held *a lot* of extra office hours.) So, I started building course projects, and my role transitioned from carpenter to tool-smith. The projects I built have fundamentally changed Berkeley's artificial intelligence course. The course content includes a very broad set of techniques, moving on to a new type of problem every few weeks. To seasoned researchers, these problems are all related. To most undergraduates, however, the course seems to jump haphazardly among a grab-bag of unrelated topics. My projects unified the course around a recurring scenario: playing Pacman.

Today in CS 188, students build agents that control Pacman in a series of five programming assignments. Each project is built around a different variant of Pacman, and thus requires different artificial intelligence techniques. While the projects are all distinct in technical content, they are consistent in many other respects. For instance, all the projects share a common agent interface, and each description begins with a haiku. Pacman has been a big hit with students: the theme taps into the students' enthusiasm for games, the graphics let students quickly visualize the output of their programming, the projects include both clear instructions and opportunities for creativity, and the scaffolding we distribute allows students to focus exclusively on the most interesting parts of software development. The projects conclude with a final contest—a multi-player, multi-agent, capture-the-flag Pacman extravaganza—that requires integrating many AI techniques from the course. The contest culminates in a "play-off" event that spawns hours of shouting, cheering, and frantic code editing as students pit their programs against one another's to determine a winner.

Responses to these projects have been overwhelmingly positive. In response to the survey question, "What worked best for you in this course?" 65% of students listed the course projects. Enrollment in the course increased a dramatic 69% the semester after Pacman was first introduced. Course popularity has continued to grow, along with student ratings about the usefulness and effectiveness of the course. Now, Pacman is spreading beyond Berkeley. Faculty at the University of Pennsylvania and New York University have used the projects for their AI courses. Last spring, we held an intercollegiate tournament with the University of Utah; Pacman plays a central role in the AI course there. At least five additional courses around the world started using these materials, even before we officially released the code to outside institutions. In recognition of these projects, I received Berkeley's Teaching Effectiveness Award, only the second time a computer science student has ever won this university-level award.

Last spring, I transitioned roles once again: I received the exceptional opportunity to serve as head instructor for CS 188—the first graduate student to lead the course in its 43-semester history. I was now the foreman, charged with using the tools I had built and teaching skills I had developed in order to build AI knowledge in 95 students and a handful of auditors. Directing a team of three teaching assistants and four undergraduate graders, I created all tests and written assignments, managed the programming projects, assigned all student grades, and delivered 27 lectures.

Lecturing has been my single favorite form of teaching. I delight in the privilege of introducing great material to enthusiastic students. Lecture also holds its own engaging challenges. Keeping undergraduates interested (or even awake) for 80 minutes requires clear explanations and creative examples. During lecture, I am constantly choosing how quickly to explain each idea, when to start an example, and whether to abandon my current slide and switch to the whiteboard. When students get confused—a reality of teaching—I draw on my experience in all those extra office hours to help me discern why they lost me and how to correct the misunderstanding.

My students enjoyed the course as well. They rated my overall teaching effectiveness as 6.4 out of 7, compared to a historical mean for the course of 5.2 with standard deviation of 0.7. My precision in answering questions was rated 4.8 out of 5; my enthusiasm was rated 5 out of 5. Students had the following to say in an anonymous end-of-semester survey:

- "*John was very engaging as a lecturer—it's very clear he will be a rock star teacher.*"
- "*Perfectly executed. Great curriculum. Lectures were phenomenally clear—and enjoyable!*"
- "*Great material, great assignments, great (fair) policies, sublimely great instructor! The best instructor I've had at Berkeley, in every aspect.*"
- "*The projects were a blast, the material stimulating, and the midterm was fair. I can't praise John or the course highly enough.*"
- "*Oh Yeah! Best CS course ever.*"

Teaching has been a great love of mine throughout graduate school. Focusing on a single course has been an ideal way to develop myself as an instructor, but I now look forward to building new courses and developing more tools for them. I expect to continue finding new teaching challenges and working with more great students, and I know I will remain an enthusiastic instructor for many decades to come.