Interactions with students have provided some of the most meaningful and rewarding experiences of my academic career thus far, and the desire to teach and mentor is a major motivation behind my pursuit of a faculty position. My background makes me well-qualified to teach courses on a variety of computer graphics topics, as well courses in artificial intelligence and programming languages. I am interested in developing new courses that straddle the boundaries of these and other disciplines, especially courses on probabilistic programming, procedural modeling, and the use of probabilistic inference and machine learning in computer graphics and design.

At Stanford, I have served as a teaching assistant for both the introductory computer graphics course (CS 148) as well as the advanced graduate course on physically-based rendering (CS 348b). In these roles, my responsibilities ranged from designing new assignments to giving lectures to developing online course infrastructure. At UC Berkeley, I was also a teaching assistant for the introductory graphics course (CS 184), where I focused much of my effort on running small group discussion sections. I also served on the teaching staff for UCBUGG, an entirely student-run course on the principles of modern computer animation, where I developed lab materials and ran tutorial sessions.

**TEACHING PHILOSOPHY**

Ownership: In my experience, students feel more engaged with and more pride in their work when they feel ownership over it. Thus, in the courses I teach, I prefer programming assignments that permit multiple implementation possibilities. As a teaching assistant for Stanford’s CS 148 introductory graphics course, I designed a new raytracing assignment along these lines, providing students with a 3D scene description format and parser but allowing them to design the rest of their code however they saw fit. Given this challenge, students in office hours and our online discussion forums engaged in productive discussions about the tradeoffs between different object-oriented design strategies. For the same reason, I also prefer courses that feature open-ended final projects. Two classes that I have been involved in teaching, CS 184 at UC Berkeley and CS 348b at Stanford, have done so. In both cases, the teaching staff provided students with suggested project ideas, but we also strongly encouraged students to develop their own ideas, helping them in office hours to flesh out and appropriately scope such projects. In addition to fostering a sense of ownership, this process helps students develop an intuition for what constitutes a significant-but-doable contribution, which is a useful skill for engineers, researchers, and entrepreneurs.

Collaboration: Most courses should involve some amount of group work, especially those with larger, term-end projects. In my experience, nearly all real-world work requires group efforts, whether in engineering or in research, so it is critical for students to learn effective collaboration skills. For group projects, I like to require project proposals that ask each group to clearly define who will work on what before the work actually begins. Collaboration can also take the form of peer teaching and learning. In the small discussion sections I led as a CS 184 teaching assistant, and throughout my time teaching UCBUGG, I saw the value of having students learn from each other. Teaching new material to others helps students solidify their own understanding. Students are often more comfortable asking
questions of their peers, whom they see as less intimidating than a professor. And practicing peer-to-peer learning in school is excellent preparation for the workplace, where official ‘teachers’ rarely exist.

**Communication:** The ability to communicate the results of technical work is an important skill: scientists have a responsibility to disseminate their results to the public, engineers must concisely summarize how a complex system works, and entrepreneurs must effectively pitch their innovations to users and investors. I will make it a priority, both as a teacher and research mentor, to develop my students’ communication skills. I have often been frustrated by the lack of emphasis on communication in Computer Science curricula. In courses I have taken, project reports and presentations have often been treated as obligatory milestones. My courses will include active instruction on how to present work in both written and oral form. This includes empathizing with the audience (“To whom am I presenting, and what do already they know?”) and establishing concrete communication goals (“What key points should the audience take away?”). I am particularly interested in exploring peer evaluation of written report drafts and practice oral presentations in my courses, something that has been a fixture of my research lab at Stanford.

**EXAMPLE COURSES**

**Introduction to Computer Graphics (intermediate):** A first course in computer graphics. Students will become familiar with fundamental graphics concepts, including geometry representation, transformations, simple reflectance models, and basic raytracing. They will also be exposed to the breadth and diversity of computer graphics via short units on topics such as image manipulation, color spaces, and digital photography. The course is strongly practical, featuring multiple short programming assignments and an open-ended final project. This class design is based on my experience TAing CS 148 and CS 184 at Stanford and Berkeley, respectively.

**Interactive Computer Graphics (intermediate):** A second course in computer graphics, focused on building usable, interactive graphics applications. Students will learn the fundamentals of real-time rendering pipelines and become fluent in programmable shading languages. They will be exposed to different methods for camera control, animation, and content creation, including procedural techniques such as noise and fractal terrain. The course culminates with the presentation of a long-term group project to build an interactive visualization, design tool, or game using WebGL. This class is based on my experience TAing CS 184 at Berkeley, as well as my experience building WebGL applications for my own research.

**Introduction to Artificial Intelligence (intermediate):** A first course in AI. Students will learn how to pose search problems and to solve them with exhaustive and heuristic techniques. They will learn the fundamentals of probabilistic reasoning and Bayesian inference, including graphical models for tasks such as spam filtering. The course will also cover basic control problems, including Markov Decision Processes, reinforcement learning, and applications to robotics. Finally, students will be introduced to basic machine learning via logistic regression, simple neural networks, and applications to computer vision. This course is based on AI courses at Berkeley and Stanford (CS 188 and CS 221, respectively), as well my experience applying search and probabilistic inference algorithms for my own research.

**Research Topics in Probabilistic Programming (advanced):** This advanced course covers both foundational and recent work on probabilistic programming languages. Students will read and discuss seminal and new papers from the probabilistic programming literature. They will also complete a long-term research project in the area: motivating and building a new probabilistic language, improving inference algorithms in an existing language, or improving the computational efficiency of an existing language. The course aims to attract and facilitate collaboration between students of artificial intelligence, machine learning, programming languages, and compilers.
Probabilistic Modeling and Inference for Computer Graphics and Design (advanced): This advanced course covers recent work at the intersection of probabilistic modeling and computer graphics. Students will read and discuss papers that apply probabilistic methods to 3D object modeling, environment modeling, color selection, surface appearance, lighting, and animation. Accompanying this discussion, students will prepare peer presentations to explain the mathematical methods used, such as graphical models, probabilistic grammars, and Monte Carlo sampling-based inference. The course culminates with an open-ended project in which students develop inference-based machinery for a novel graphical design tool.