

Teaching Statement for Chris Wyman

The primary mission of every college and university is to provide a high quality education for all of its students. This mission requires every professor to challenge and engage his students, striving to clearly convey information so as to stimulate and excite all students in the classroom, not just those at the bottom, in the middle, or near the top of the class.

To accomplish this goal, I believe a teacher must excel at three major pursuits:

- Clearly presenting relevant material.
- Exciting students about the coursework.
- Teaching *how* to learn, not focusing on just a few facts.

My Experiences

My undergraduate and graduate teaching experiences will be invaluable to my future teaching responsibilities, as they have shown me how to approach these three goals. As an undergraduate at the University of Minnesota, I spent two years as a teaching assistant. I graded homework, lab assignments, exams, and held regular office hours. I also led recitation sections, which included clarifications, reviews, discussions, and mini-lectures. For five of the quarters I was a TA for CS 3316, an introductory computer science course using the Scheme language with 250–350 students. My last quarter as a TA was for CS 5107, an introductory graphics course for third and fourth year undergraduates as well as graduate students.

Due to the research fellowship I accepted as a graduate student, I have not held any official teaching positions at the University of Utah. However, I have on numerous occasions filled in for my advisor's graphics class (CS 5510/6610, a graphics course discussing advanced OpenGL rendering techniques), including 4 weeks while he recovered from surgery. This included preparing lectures on a wide variety of graphics topics, proctoring and grading exams, and holding office hours by appointment.

For three summers as an undergraduate and graduate student, I was a teaching assistant at an astrophysics camp for bright high school juniors and seniors called the Summer Science Program. The curriculum at the program included college-level mathematics, physics, and computer science. My responsibilities at the program included grading papers, holding impromptu review sessions, answering questions, and supervising other teaching assistants. I also designed the program's introductory computer science curriculum. In six three-hour lectures, I taught enough C programming for students to write a program to compute the orbit of the asteroid they observed over the summer. Many of these students had never programmed before, and some had never used a computer.

Teaching Philosophy

My teaching experiences have shown me that each student learns differently, so clearly presenting material requires effort beyond simply knowing the material. Some students learn best through hands-on projects, others through attending lecture, completing homework, reading the book, or asking one-on-one questions during office hours. Good courses require interesting and relevant projects and homeworks, high quality books, understandable lectures, and answering one-on-one questions with patience. Also, it is important to gauge your class. Graduate courses are different than upper division courses, which are different than an introductory class. For graduate courses, selecting relevant homework and projects is perhaps most important, whereas introductory courses need crystal clear presentation both in the lectures and supplementary materials.

Exciting students about the work keeps them involved in a course and encourages them to explore the field beyond the scope of a single class. I have noticed that students who lack enthusiasm for a course tend to be those who miss class, turn in work late, and avoid asking for help. Keeping students excited can be challenging, especially for those

who merely take the class to fulfill a requirement. However, by remaining animated in class, providing interesting and useful assignments, and most importantly conveying my own excitement for the material, I believe I can excite students.

I believe the most important lesson one can learn is *how* to learn. Facts, techniques, and algorithms often grow outdated. If students fail to learn the importance of lifelong learning, they become lost as their knowledge and skills grow outdated. Computer science is a young field which evolves quickly, so keeping up with the field requires constant learning. As no clear way exists to teach someone how to learn, my approach is to challenge and encourage students while avoiding pitfalls which prevent the process. Exciting students about class material encourages them to learn beyond the scope of a single course. Presenting coursework clearly prevents them from becoming discouraged and convinced that the course is “above” them. Additionally, by remaining up to date myself I can avoid starting students at a disadvantage, and can pose questions which challenge students to think about modern problems and how to solve them.

Teaching Courses

Given my teaching experience and knowledge of computer science, I believe I am qualified to teach any core computer science course at the undergraduate level. Obviously, my research experience and knowledge will be most applicable to graphics coursework. However, many areas of computer science interest me. Below I outline courses and topics I feel best qualified and most excited to teach.

- **Introductory Graphics**, including fundamentals of computer graphics (e.g., linear algebra), coordinate systems, basic shading models, rasterization, ray tracing, and graphics APIs.
- **Advanced Graphics**, including advanced shading models, perceptual issues in graphics, acceleration structures, texturing, procedural techniques, image-based techniques, and basic visualization techniques.
- **Interactive Graphics**, including advanced OpenGL techniques, efficient data structures, fast illumination methods, data representation for interactive rendering, and graphics hardware.
- **Computer Graphics Mathematics**, including linear algebra, probability, numerical integration, and Fourier, wavelet, and spherical harmonic transforms.
- **Computer Vision and Image Processing**, including edge and feature detection, Hough transform, morphing, filtering, image segmentation, and image compression.
- **Artificial Intelligence and Machine Learning**, including searching, logic, neural networks, Bayesian learning, genetic algorithms, and natural language processing.
- **Theory of Computations**, including mathematical logic, automata, Turing machines, decidability, and P/NP issues.
- **Introductory Computer Science Courses**, including introduction to computers, computing for engineers, discrete math, data structures, and introductory programming courses using languages such as C/C++, Java, Scheme, BASIC, and Assembly.