William E. Byrd—Teaching Statement

1 My Teaching Style: Focused on Students and their Interests

> There are better writers than me out there, there are smarter writers, there are people who can plot better—there are all those kinds of things, but there's nobody who can write a Neil Gaiman story like I can. —Neil Gaiman

There are better teachers than me out there, there are smarter teachers, there are people who can program better—there are all those kinds of things, but there's nobody who can teach a Will Byrd course like I can.

A Will Byrd course focuses on the interests of the students to the greatest extent possible. These interests change every semester, of course, so my first and most crucial task is to get to know every student. I start by asking students to fill out a short questionnaire on their background, interests, and goals for the course. I then use a "virtual flashcard" program to quickly learn the names and backgrounds of the students. (In an introductory programming course, this flashcard program itself provides a learning opportunity, as an early example of a simple but useful program.) Most importantly, over the first few weeks of class I invite students to lunch in groups of three or four. These lunches are also a terrific way to get students' feedback and suggestions for the course. Throughout the semester I continue to invite any interested students out to lunch in small groups.

As I get to know the students and their interests, we begin discussing a *project theme*, which will guide the specific assignments and projects we will explore during the course. This discussion takes place over two or three weeks, during which time we cover fundamental material necessary for any conceivable project theme. In my Fall 2010 section of H211[1] almost all my students—men and women—were interested in games and game design; as a result, I based all but one of our projects on games. (The remaining project was fractal movie generation, which was also extremely popular.) The following Fall my H211 students were less interested in games, but almost every student played a musical instrument. For our projects we generated music procedurally using Markov models and built Arduino-controlled polyphonic touch-sensitive electronic pianos; my students' capstone projects included an Arduino-controlled robotic flute that uses compressed air and servo motors, a glove-based synthesizer that automatically transcribes the music being played, and a 3D-printed glowing jellyfish that "dances" to the beat. I make a special effort to learn about the interests of any students who are not as enthusiastic about the project theme, and to arrange for supplemental activities and assignments that they will find appealing.

To help foster a sense of community, and to encourage a sense of humor and playfulness when learning difficult material, each class decides on a *class theme* (not to be confused with the project theme). For example, the Fall 2010 section of H211 chose Joss Whedon movies and TV shows for their class theme. This theme showed up in everything from variable names to an extensible database of *Firefly* trivia integrated into one group's capstone game. In Fall 2011, our H211 theme was *My Little Pony: Friendship is Magic* (which rapidly spread throughout the computer science department, to the delight and dismay of students and faculty). A few of my other community-building efforts include game and movie nights, a student-run Facebook page, and class reunions

[1] H211 is Indiana University's honors introductory programming course.

after the semester ends. As a result of these efforts I keep in touch with a surprisingly high percentage of my former students, which in turn leads to daily discussions of computer science, school/career choices, and countless other topics, with students I haven't seen in person for years. These conversations have given me a very long-term view of teaching and mentorship, and help remind me that most learning takes place outside of the classroom.

Other aspects of my teaching philosophy that I have developed over many years include:

- The importance of continuing education for a teacher.
  It is a student's right to be taught by someone with real expertise in their subject.
- The importance of properly motivating students.
  Learning something interesting or surprising is inherently motivating.
- The importance of projects with universal appeal that students can show off to friends, parents, and grandparents: fractals, animatronics, digital fabrication/3D printing, computer-controlled musical instruments, etc. *Bling is good, bling-bling is better.*
- The importance of ambitious projects that push the skills and creativity of students. *We will succeed epically or fail epically.*
- The importance of embracing mistakes, and of ensuring that students have enough chances to make mistakes. *An expert is someone who has made more mistakes than you have.*
- The importance of semi-structured activities in which students can learn from each other. *In the words of FIRST robotics hero Woodie Flowers, "Learning with always trumps learning from."*

A (regrettably low-fi) recording of a talk I gave on my teaching philosophy, after I first integrated Arduino into the H211 curriculum, can be found at `http://vimeo.com/23254348`.

2 Undergraduate Research

I have been fortunate to work with many outstanding undergraduates over the years. Dan Friedman and I wrote several papers on logic programming and functional programming with Joseph Near (now at MIT), Ramana Kumar (now at Cambridge), and Claire Alves (now at Northeastern) while they were undergraduates at Indiana University. Claire wants to continue the work we did on constraint logic programming as her PhD topic.

Two summers ago I worked with three of my best former H211 students—Emilie Mitchell, Tevyn Bell, and Brittany Moore—on grammar-based jazz analysis, based on the work of Harvey Mudd professor Bob Keller. This project was largely an experiment to see what would happen if I got first-year students (Emilie and Tevyn) involved in research, instead of waiting until their junior or senior year. Accordingly, we spent the first part of the summer learning background material (such as reading through and implementing much of Michael Sipser's *Introduction to the Theory of Computation*). The students also attempted to teach me music theory (which, alas, was not as successful). We were not able to implement the entire jazz analysis algorithm before the summer ended, but the experience seemed to have been beneficial for the students. Emilie was inspired to became a music/computer-science double major, and all three students

found the advanced functional programming techniques I taught them useful in their classes the following school year. All three students have also been interested in continuing with undergraduate research: I am currently working with Tevyn on a neurosymbolic computing project, combining logic programming and neural networks; several months ago I supervised Brittany's research on rule-based interactive fiction; and Emilie and I are talking about doing a machine learning project for automatic transcription of hand-written musical scores.

3 Arduino, Digital Fabrication, and CS Education

For the past four years my hobby has been building robots that build robots.

The availability of low-cost, easy-to-use microcontrollers, such as Arduino, along with the appearance of affordable desktop 3D printers, means that it is easier than ever to design, build, and program robots and animatronic creatures. In a manner reminiscent of bootstrapping a compiler, it is now possible to program an Arduino to control a 3D printer, then use the printer to print out parts for a new 3D printer, which in turn is controlled by an Arduino. As with bootstrapping a compiler, the new printer can be *better* than the printer on which it was created. I find this prospect endlessly fascinating, as do many of my students.

I am interested in how digital fabrication, 3D printing, and Arduino can be used to help students learn programming and computer science in new ways. After two years of integrating Arduino into H211 at Indiana University, I designed and taught a new course on physical computing at the request of the computer science department. In addition, I helped with the electronic textiles workshop at the 2010 Indiana Celebration of Women in Computing (InWIC), and helped two of my former H211 students, Maria Khokhar and Brittany Moore, plan a 3D printing demo that won the "People's Choice" award at InWIC 2012. Maria and I are building an Arduino-controlled Segway clone (a "Seg-bot"), which we work on whenever I visit Bloomington.

With the help of my former H211 students, I started informal Arduino/physical computing and 3D printing clubs, which have now been combined into an official student-run club, the "Bloomington Fabrication Group." Club members build and operate Open Source Hardware Arduino-controlled 3D printers, design and model printable musical instruments, and control those instruments using servo motors, solenoids, and Arduino. In 2011 Mark French, a professor in Purdue's Mechanical Engineering Technology (MET) Department, invited me to speak about Open Source Hardware 3D printing; based on my visit, the MET Department created an undergraduate digital fabrication club and purchased kits for several 3D printers and computer-controlled routers.

During my trip to Purdue, Mark French and I visited an elementary school in Lafayette where I demonstrated 3D printing to a third-grade class; the students were mesmerized by the 3D modeling process, the operation of the printer, and the final printed objects. Most popular was the "cat soap" I made by 3D scanning my father's sculpture of our family's cat, 3D printing the model, vacuum-forming a mold using the printed piece, then filling the mold with melted soap. The girls especially loved the cat soap. I think there is a fantastic opportunity for undergraduate students to use digital fabrication technologies to show girls how science, engineering, and CS can be used creatively and artistically.

In 2011 I led a multi-week Arduino animatronics workshop at the Bloomington Boys & Girls Club. The workshop was especially popular with middle-school age girls. I am especially interested in teaching teens to use digital fabrication technology, including 3D modeling and 3D printing—I would love to involve undergraduates in teaching digital fabrication at a local branch of the club.

4 Other Course Ideas

In addition to standard core CS courses, courses related to Arduino and digital fabrication, and specialized courses on relational/logic programming, functional programming, and programming languages, I would like to develop a "retro programming" course as a way to get students closer to the hardware. I would love to teach an Atari 2600 or classic Nintendo Entertainment System (NES) game programming course, somewhat similar to the Atari 2600 course Ian Bogost teaches at Georgia Tech but using 6502 assembly instead of a high-level language. The 6502 could also serve as the target architecture for a compiler course. Partly to prepare for such a class in the future, my friend Andy Keep and I have started a 6502 Retro Game Hacking Group at the University of Utah. After several months of working with students, we have a crude but working prototype of what we believe to be the world's first networked NES game.

I am also interested in how games can be used in CS education. I am currently advising one of my former H211 students, Brittany Moore, on the design of Weaver, a declarative programming language for Interactive Fiction (IF) based on defeasible logic. In addition to involving other students in this research, I would like to develop a course on IF. This course would appeal to students who enjoy writing and telling stories.

Although I rarely have time to play, I absolutely love the StarCraft family of real-time strategy games, which have been described as "chess for the highly coordinated." I would like to form a student team to enter the annual AIIDE StarCraft AI Competition[2]. UC Berkeley has used this competition to teach AI to undergraduates and as a fertile ground for research, leading to the Berkeley Overmind project[3]. If this experiment is a success, it could lead to a course that uses StarCraft to teach various ideas from AI—for example, students could create a build-order optimizer using genetic algorithms or constraint logic programming.

[2] `http://webdocs.cs.ualberta.ca/~cdavid/starcraftaicomp/`

[3] `http://overmind.cs.berkeley.edu/`