2014-06-18-kim-voll-notes

June 18, 2014

Contents

1 Kim Intro

Dr. Kimberly Voll is senior faculty at the Centre for Digital Media, where she teaches Master's students various topics related to interactive digital media. Kim's has a Ph.D. in computer science, specializing in AI, game design, usability, and cognition.

2 Kim Topics of Special Interest

I think I can especially speak to applied learning and my own experiences in that sphere. And of course gender issues in CS, and video games, and virtual reality.

But generally I'm happy to go where the wind takes us! So ask whatever you like.

3 Kim Question Pool

- What's the most interesting non-CS teaching experience you've had? What did you learn from it?
  - What's something you're a student of now? What does your experience as a student tell you about teaching?
  - How have those experiences informed your teaching in CS?
- What's exciting to you in education that you have tried?
- What's something you think every CSists should read, learn, do, or play with?
- How do you share your excitement about Computer Science with people outside the field?
- What's something we really don't need to know about CS anymore that's in our curricula?
- What's a course you wish you could teach?
- What's your favorite computing story?
- How do you establish the atmosphere of the course, where people are engaged, excited, and willing to take changes?
  - How do you fix the atmosphere in a course that's gone awry?
- Describe a teacher you've had that you admired and what you admired about them.
- What tips do you have for successfully teaching a large class (100+ people)?
- What tips do you have for successfully teaching a small class (10- people)?
- What's something cool you do in your teaching?
- What do you do to prepare before a course starts?
- How do you get to know your students?
- How do you assess your students?

- What do you do in lecture/class?
- What's exciting that you haven't tried?
- How should CS reach out to the public?
- How do you identify and help students who are struggling?
- How do you help students who are enthusiastic for extra work?
- How do you keep a course fresh when you teach it multiple times?
- How do you handle challenging students?
- How do you manage TAs, both so that the basic needs of the course are met and so that those willing are inspired and able to go above and beyond?
- How do you manage the classroom environment, particularly cell phones, laptops, and other distracting devices?
- How do you get people to participate in class discussions?
- How do you answer questions ("good" questions, "dumb" questions, etc.)?
- What's something every Computer Scientist should know that's not in our curricula?
- Tell us about something you tried in your teaching that went horribly awry.
- Do you have topics where your own interest flags? What do you do?
- What's something that students consistently have trouble learning in your courses? How do you address it?
- What do you do for students who REALLY want some topic but your school just doesn't do it?

4 Actual Questions

- Steve: What's the most interesting non-CS teaching experience you've had? What did you learn from it?

  - Steve: How have those experiences informed your teaching in CS?

- Will: How do you set up the environment of the course?
  - KIM: "I look at a classroom as a room full of equals with myself as one among them."

- Will: What are some activities you'd run and how would you structure them?
  - KIM: Break what you want students to learn up into "cognitive units" so you can have a foundation that people can build new skills on.
  - Will: Specific example of something that works really well in that style and one that you struggle with in that style.
    - KIM: Teaching proofs has always been one of the hardest things because you can show people a proof, but that's not what we want them to do. That independent generation of a solution doesn't necessarily break down into a clean-cut hierarchical sort of approach.
    - KIM: We as experts need to divorce ourselves from our "chunked" knowledge because novices don't **have** those chunks.
  - Will: How do we help the students who are enthusiastic about games? Traditional CS? Game-oriented?
    - KIM: A lot to be said for a "traditional" CS degree: all the tools are in your toolbox.
    - KIM: NUMBER ONE piece of advice. If you want to go into the games industry, you **make games**. Maybe you don't have a games course? Get over the inertia and develop a game yourself. You may be making crappy games, and that's **OK**. Just make games.
      Work on **finishing it**. Make the UI, polish the game, distribute it.
    - KIM: What do students need that they're not getting? Know how to work interdisciplinarily. Know how to talk to artists, for example. Know how to talk to and really **work with** domain experts.
    - KIM: **Dedicate** a few hours a week to a personal project.

        * KIM: A portfolio isn't an exhaustive list. It's what the challenges you faced were and how you solved them, who you worked with and how you collaborated successfully with them.
        * KIM: Try out something like Global Game Jam. A bite-sized, wholistic development opportunity.
        * KIM: Get involved with local game developer meetup groups.
    - Will: What should these top students interested in gaming be doing with their "free time" to prepare? (Handled above.)

- Steve: What should we CS educators put on our "summer reading list"? Something you think every CSist should take the chance read, learn, do, or play with?
    - KIM: Get out there and make a game. Read up on Unity (dominant engine in the game programming world). Component-based programming vs. traditional OO.

- Steve: Teach whatever you like, what would it be?
    - KIM: An applied course (probably game design!) where students have to deal with time constraints where they **cannot** do everything they want to do. Making choices about where to optimize and where to just get it working and learning how to tell the difference are all important.

- Will: What did you learn from your great teachers?
    - KIM: Creating a classroom where I was an equal.
    - KIM: Facilitating student projects (e.g., research) whenever possible, keeping the barriers **very** low.

- Chris: What do you do to help practitioners in another field who need to start programming immediately?
    - KIM: Find the foundation that they need to accomplish whatever it is they're trying to do.
    - Petey: How do we answer a question that's going to take too long to answer right now? (E.g., stopping at the Bohr Model of the atom vs. quantum theory.)

        * KIM: Help them discover what the problem they're trying to address **really** is. It may just be a matter of perspective that makes it seem like they need deep knowledge of some intricate detail.

- Will: Words of wisdom for teachers getting started?
    - KIM: Professors don't know everything.
    - KIM: My job is to create curated experiences that will help students establish life-long learning.
    - KIM: Pay attention to the different ways you and students struggle.
    - KIM: Keep an open mind; be receptive to and responsive to criticism. The classroom integrity breaks down when faculty have become too immovable.
- KIM: Get some dev kits, create opportunities for people to get together and just **make stuff** together.
- KIM: Consider inviting Phill Conrad from UCSB

4.1 Terminology

- Learning goal
- New prep
- UI
- Unity: link to this
- Pipeline
- Component-based programming
- OOP
- Bohr model (of the atom)

5 TODO list for next time

- Set up the “question contribution list” in advance so we HAVE questions when we ask for them.
- Have question-askers give a 10-second intro of themselves.
- Decide how long we let our own Q&A go before opening up.
- Consider a break in the middle (Part 1, Part 2)
- Short question, long answer; kick off with “tell us about” or “can you talk about”
  - Ask the speaker how to accomplish that “long answer” format