*"The arts and humanities are sexier than the sciences. Every kid wants to be a singer, a dancer, a rapper. Programming a computer? I think it's sexy, but I know a different side of it."*

# Pat Yongpradit

*9th–12th Grade*
*Computer Science and Technology*

*Mr. Yongpradit with students in the Computer Lab of Learning, working on a video game project.*

**When Pat Yongpradit graduated from college, he had a plan: get his master's degree at Stanford, become a teacher, live in a small coastal community, and buy a convertible. He knew California would be expensive, but figured, "As long as I had a Miata and lived near the water so I could surf, I'd be happy." Ah, the best-laid plans. Finances and common sense intervened and Yongpradit ended up getting his teacher's certification in his home state of Maryland. He now lives and works not seven minutes from where he grew up. Yongpradit is passionate about the subject he teaches: computer science. His unique approach to teaching programming is embedded with the themes of public service and real-world relevance. Instead of creating games about blowing people up, Yongpradit's students make games and apps about saving people's lives. His unique integration of ethics and computer science won him the Microsoft Worldwide Innovator Award in 2010, and recently he became the Director of Education for Code.org, a national non-profit organization promoting computer science education. Last year, he finally got a Miata.**

What I love about computer science is that you don't just read about it, you do it—it's a doing kind of subject: 10 percent of the time you might be learning from a book; 90 percent of the time you're learning from a hands-on project. And with

each project, you're creating something completely novel, because it comes straight from your imagination. No one else has ever done it the way you just did it. And, that's the beauty of it. It can be a very creative, innovative subject.

By my third year in the classroom, I started to feel comfortable. I had amassed dozens and dozens of labs and PowerPoint presentations, along with movies, assignments, tests, and quizzes; I had material prepared for every possible situation. At that point it was just a matter of cleaning it up and making it better. I chose contexts for my project assignments that kids would find fun and interesting—like wrestling or video games. That approach went well, except for one big thing: It mainly appealed to the crop of kids who would normally take computer science, like young boys. This illustrates a widespread problem in computer science: We over-appeal to a subset of the population, which leads to the stereotypes that we often see in our field.

**I've learned a lot from you. Especially about ethics. I know that's weird but you told me once that people were going to come and go in my life, but I'd always have my integrity. That was one of the most influential things anyone has ever said to me and I still think about it every day. Thank you.**

*Meredith, age 21, former student*

Then I had a professional-development opportunity through the Microsoft Worldwide Innovative Education Forum (now called the Partners of Learning Global Forum) in Cape Town, South Africa. I applied to the forum and was invited to present there. I basically put forth my best project and I placed first. They used a very detailed rubric to judge the submissions, which introduced me to a new way of understanding what an innovative project is. Through that experience in South Africa, I came to realize that I needed to think beyond my classroom—and get involved in more social causes.

I wanted to extend my approach beyond the contexts I'd been using for my lessons and my assignments, beyond the normal computer-science-y kinds of topics—even beyond the culture of computer science, which would very typically include topics like video games.

After I returned, my first assignment was to have kids model the life of a South-African street child. A *tsotsi* is like a street thug, a criminal, and unfortunately, some of these street children become tsotsis. I asked my students to program characters based on these kids. I had them really think about and research what it is like to be a South-African street child. What do they go through? What are their behaviors? What do you need to think about when you're portraying them in a computer program?

That marked a big change in how I approached my teaching. From then on, I started connecting my big student projects to the eight United Nations Millennium Development Goals. I thought, if kids are going to write video games—which I was still a fan of—they were not going to be about zombies and shooting people, but about issues surrounding poverty, maternal health, universal education, environmental sustainability, and so on.

Having to address a public need is integrated into every single project I assign. Honestly, the cleverest work I've gotten out of my students is when they have to think and develop an idea within the context of a social cause—much more so than when they were free to do whatever they wanted. For example, I had a student make a game about how to get a date with an environmentalist. It's almost like a "choose your own adventure" game, where you have to answer a female environmentalist's questions. She quizzes you, and you have to give the correct response, otherwise you'll lose the opportunity to get to know her better. Another group of my students created a game about maternal health called Just in Time, in which the main character is traveling around a rural village providing mothers in need with vitamins, food, and medical transportation. Another game they invented was called Electra Hope, which was dedicated to bringing attention to environmental sustainability and the energy crisis in the aftermath of the Tōhoku tsunami. My students have also built an app for the DC Office of Homeland Security, created games in which kids use their body to solve math problems, and just recently came up with an app to describe the anatomy of a cluster bomb in partnership with a nonprofit called Legacies of War.

I believe in making everything I teach applicable in the real world, and not just in terms of the content, but in terms of the process, the pedagogy. For instance, I have my students use industry practices while they're coding, like pair programming. Rather than just one person programming by him- or herself, you have two people working together at the same computer, taking turns with the mouse and keyboard. The person using the mouse and keyboard is called the driver and the other person is called the navigator. So imagine you have two people in a car. One person has the map and is directing the other person where to go, so the driver can just concentrate on driving. Instead of working the way most people think a computer scientist works, which is alone, the kids are constantly interacting socially. And instead of asking me for help when they need it, they turn to their partner for help. So there's this constant collaboration. If you could ever see a video of my class when pair programming is happening, it's very buzzy, very dynamic. The kids are all engaged. They might be laughing. But they're also correcting each other, thinking of new and better ideas. I love that kind of environment.

The whole change in my approach to content and pedagogy coincided with this ambition to have my students enter as many contests as possible. Not because of the notoriety it brings us, though that does help to attract kids to the program, but more just giving kids a reason to expand their education outside of the boundaries of school. These are competitions for games about social causes. Like, the Kodu Cup. It's all about creating a game to represent the relationship that people have with water. So my students are making games about water pollution and water scarcity. If outside organizations are going to present competition opportunities like that and offer thousands of dollars in awards, then I'm going to shape my projects to match up to them. It's win-win.

I want my students to understand the value of competition. I think that's a really important component that's missing in American education. We all believe in the American dream, but the fact is you must compete to attain the American dream. Everyone can get an A in school, but there is always a better writer, better scientist, a better whatever. I'm not a competition monger but these contests allow me, as a teacher, to tell students, "If you do this, this, and this, then you'll get an A from me, but you're not going to win the contest." I can even ask them right off the bat, when they're designing their game or their project, "Do you think this is going to win? Is this a winning idea? Because if it's not your best idea, I don't want you pursuing it. I don't want you spending the hours and hours it's going to take to create it."

Students often don't understand the difference between fulfilling requirements and doing the best they can. They haven't made that mental switch. And that's why I do these contests: to teach them how to make that switch. And they've done very well. They've excelled at it. We participated in a national contest recently in which there were awards for the people's choice and the judge's choice in the game design and app design categories. Springbrook teams won both awards in both categories. They swept the entire competition.

*"Computer-science education is my love. I really believe in it, and I want everyone to experience it."*

Since I started teaching at Springbrook, we've doubled the size of the computer science program. It's taken a lot of hard work. We compete for new students. Some people don't like that, but I think it pushes programs to be better. I'll go out and talk to middle schools. Middle schools will come to us. We're trying to attract the best and the brightest, the kids who really care about technology. Kids hear about the program. They see the results. They see what our students are doing when they graduate—the internships, the jobs they have lined up, the college scholarships, the actual applications and games that they've already sold and are out in the marketplace. And they want a piece of that. The natural result has been that the program has grown. Everything has doubled—the number of girls in the program, the number of classes overall. Everything.

In terms of where computer science stands in the American education system, there are only thirteen states that consider the subject credit-worthy: where you can take a computer science class and it will actually count as a credit toward graduation. In most states, including mine, it's like a pure elective. It's not even on the level of yoga. With yoga, you get a PE credit. People don't understand its place—which is crazy when you think about all the innovations in technology during the last twenty or thirty years. When it comes to anything related to science, technology, engineering, or math, computer science is like half the pie: Half the job growth annually is in computing jobs. But the arts and humanities are sexier than the sciences, so kids gravitate toward those. Every kid wants to be a singer, a dancer, a rapper. It's what's on television. It's better publicized. It seems more fun, which I totally understand. Programming a computer? *I* think it's sexy; I think it's wonderful. But I know a different side of it.

Computer-science education is my love. I really believe in it, and I want everyone to experience it. Even if they don't think it's for them, which is completely fine. I just want them to have a shot.