Infusing Joy into Computer Science: How to Get 65% Female Enrollment

Host: Donna Milgram, Executive Director, Institute for Women in Trades, Technology & Science

Presenter: Dr. Dan Garcia, Teaching Professor, Department of Electrical Engineering and Computer Sciences, Developer of the "Beauty and Joy of Computing" Course, University of California, Berkeley

Interview Transcript:

Donna: Hello and welcome to the *STEM Success for Women Telesummit* funded by The National Science Foundation. In our first session, we heard from Mark Evans who used drones and Spheros robots along with other fun strategies to engage prospective students. Our next guest, Dr. Dan Garcia of UC Berkeley is nationally known for proving that computer science can in fact be joyful. My name is Donna Milgram, Executive Director of the Institute for Women in Trades, Technology and Science, and I am so excited that you could join us for this online conference. We've brought together 15 speakers and over a thousand educators and we'll be showcasing real life, in the trenches educators and change makers who are making a difference in their schools, in their classrooms and in the communities of STEM and career and technical education.

Dr. Dan Garcia

Our guest today is Dr. Dan Garcia, a Teaching Professor in the Electrical Engineering and Computer Sciences Department at UC Berkeley. **His "Beauty and Joy of Computing" (BJC) course, at UC Berkeley, recently shattered campus computer science diversity records with 65% female enrollment in Spring 2018.** Selected as an Association for Computing Machinery Distinguished Educator in 2012, he's won all four of the department's computer science teaching awards, the National Center for Women & Information Technology (NCWIT) Undergraduate Research Mentoring Award, the UC Berkeley Unsung Hero Award, the SAP Visionary Member Award, and the Level Playing Field Institute (LPFI) Lux Award for his work to diversify computing. His team has offered professional development to over 400 teachers nationwide. Welcome Dan, and thank you for joining me for the *STEM Success for Women Telesummit.*

I remember when I first heard about this introductory course at UC Berkeley, called the "Beauty and Joy of Computing," and how it was attracting a huge number of female students and I thought, "How brilliant." The title of your course especially. It re-frames computer science from something just for nerds and insiders to something attractive and fun for everyone. So, I'm eager to hear you share how you came up with the idea for this course and what are the key principles that set it apart? My understanding is your 65% female student enrollment is among the highest percentage of female students at a university intro computing course in the entire country, and you've joked about needing to reach out to male students. So I'm wanting to ask you, what your computer science course is doing differently to appeal to so many female students, so that the many educators that are listening can replicate your successes into their own courses. **Before we dive into the principles, can we just talk a little bit about the course title and what you did to start to recruit women?**

Dr. Dan Garcia (Dan): I'm delighted to be here. This is just wonderful to be able to share some of the things we've been doing with other educators. The course title is a funny story. In 2005, the computer science field was suffering from folks fleeing the field. It was the funniest thing, at the time when computing is everywhere. Cellphones were just in everyone's hands, people weren't going into the major. Grady Booch came and gave a keynote address at the SIGCSE, that's the Special Interest Group Computer Science Educators Conference. It's kind of an annual conference that all CS educators go to. He said, "What's missing about today's computer science course is that, that was there when I had it, what's missing is the idea of the passion, the beauty, the joy and the awe of teaching and making a computer do what you can't, maybe creating artificial words that weren't there before."

We started a series of talks called, "How to bring the passion, beauty, joy, and awe back to computing?" at the SIGCSE Conference. Every year, I was moderating this and bringing some speakers in and that phrase, “passion, beauty, joy,” came in. As we were thinking of redoing our big non-major's course, we said, "Let's call it that." We thought about it and we said, "Passion is something that's inherent to the person. You don't teach passion, so you can't do that one." Richard Karp, who's the Turing Award winner of our department said, "Awe doesn't have the right meaning because it's shock in awe and awe is something you look at and you can't really touch," so we said, "Just keep it simple and go with beauty and joy." And we said, "That's it! The Beauty and Joy of Computing." And we were sold.

Donna: I love that title, and here we are in 2018, it's sort of hard to imagine that there wasn't interest in computer science. Things have changed so much. So I will say, for females interest has really gone down, and there's statistics to show that in computer science, but yet with regards to your course the interest has gone up, which is why you're here. **So because there wasn't a lot of interest back in 2005, how did you recruit for the course in the beginning?**

Dan: Sure, well what we had done is as I said ... We had this course, it was a text based language and it was all about computer science. It was really our intro course CS1, our first course for majors. And we kind of took the first six weeks and stretched over 15 or 16 weeks. So it really wasn't anything other than what the normal majors were getting. We said "Let's rethink the whole thing. Let's think about what courses should be ..." And by the way, I need to say, all of this came from October 26, 2008. Jan Cooney at the National Science Foundation, she had a major summit that brought all the CS educators together to think about could we have a new non-majors, both university courses, and could we have a course at the AP college board level. So that was the meeting that got me started, so I need to give credit where credit is due. Jan really got the ball rolling.

Again, 2008, I ran back to Berkeley and said, "We need to rethink our course." And so we literally had four whiteboards, and we had my colleague Brian Harvey and grad students and undergrads who were wanting to do this. It was really a fun design process of how we could take the best things from our old course, preserve those, and then swap out all the things that weren't working and just really make a chance to change everything, so that began the design process. And what we really did was start very small, when we first finished the first version of the course, we were moving from a text-based language to a graphical language. We looked across the country and we saw that Scratch was an amazing language, the best interface, the blocks-based model of programming was really the way we wanted to go.

But it didn't allow us to write functions. We had to be able to write functions that could encapsulate how to do something, give it a name, give some arguments to it and do all that. They couldn't do it at the time. But there was a programmer named Jens Mönig who had a version of Scratch called BYOB, Build Your Own Blocks, and we said "Well let's try that." So we all made the decision to keep the course really small initially, so we didn't try to open up to everybody, it was actually in the pilot stage for two full years. So the first year we only allowed 16 students in so we could really get to know them. We actually had like eight staff, and the students worked paired programming, so literally every pair had a person working with them making sure they were having no trouble and making sure the software held up, because we really didn't know if the software could hold up.

So that became 16, the next semester we went back to the drawing board and really polished it up so we didn't offer the semester. Then we came back and we said, "Let's open it up to as many people that could take it." And no one knew about it, the word hadn't gotten out. So it went to 80, and then 160, and then 240, and then 300, and now we're kind of at a steady state of about 500 a year, so we're really delighted about those numbers.

Donna: Wow, now I'm curious, when you were piloting it, did you have half women in that initial 16?

Dan: Actually the funny thing is we did, I think of the 16 we had 7 women. So we were in the range between I'd say 35-45% all throughout. All throughout the entire process of all the ten years now almost we've been doing this. And the funny thing is, we wouldn't be having this phone call if it weren't for a keynote I gave at a conference at Association for Computing Machinery (ACM) Richard Tapia Conference for the Celebration of Diversity in Computing, and I mentioned that we had just, the Spring of 2013, just eclipsed 50% enrollment in women. I asked the campus to pull the numbers and it had never happened before. So that was a big deal. And it got tweeted, this is the funny thing, it got tweeted as "Berkeley is now 50% women." That's the incorrect tweet, as a department we're not, as a major we're not, but that particular course we are.

But because the tweet seemed to imply that the whole department was, then everyone was calling. So I'm getting calls from everybody, the front page of the San Jose Mercury News, and I corrected them every time. Whenever they'd ask me I'd say, "That's actually not what it is, here's what it is." But once we got on the phone they still wanted to hear the story, so that got a bit of national exposure, and what's wonderful is as that national exposure kind of bubbled down to people telling their friends, and I would always tell the students tell your friends this course is better. There's a lot of project work in the course, so I would say, "Tell your friends so they can come and you can have a better experience in the projects."

That word of mouth really was the thing that got it rolling, it was really interesting. Once that had continued, the numbers continued to grow and next thing you know we hit 50%. And then what's funny is I didn't do any more recruiting, it was kind of word of mouth. I did my thing and I was getting some national exposure so maybe they were seeing that. The next time I looked, now we're at 60% and the next thing I look this semester we were at 65%. So I haven't done anything other than do the same thing. There's a lot of other things I've done, but I haven't done any explicit recruiting, I just basically let the course sell itself. I think that's the best thing. It's best when students have really been engaged by the course.

Every year I have students come up to me. In fact, the second year, students came up to me one after the other and just tugged on my shirt and said, "Dan, can I tell you how this course has changed my life?" And I'd just gotten a cellphone that had a video camera, so I said, "Can I put you on video for this?" And they'd say "Sure." So I went to the neighboring classroom and I recorded their story, and story after story came that particular exam. It was a hard final exam I don't know why they weren't angry at me and come at me with a sock full of quarters, but they all wanted to tell me how this course changed their life. I put those all on our front page, there's a YouTube playlist that houses it. And that might've helped, to have students tell in their own words why this course meant so much to them. So really if you're going to do recruiting, let the students sell it. Because it really is meaningful having someone else say "Come try this!" It's best to have the people who have experienced it and really have had it make a difference for them sell it for you.

Donna: I heard a couple different things. One is, of course, the testimonials from the students themselves, which you videotaped and are now helping to recruit and you have on your department's page. The other is that you've got some social media press, you've got both social media via Twitter and some press, and that helped as well. But the most important thing that I heard I want to go back to, which is you didn't just take the course that you already had and give it an attractive name, you actually fundamentally changed the course. And you did it over two years. You piloted it, you had female students as part of that pilot, so you were getting that feedback in the loop to any modifications you were making. **Can you talk about what kind of changes you made to the course and why you made them?**

I know that you and I have discussed that you have a number of different principles in terms of changing the course content and pedagogy, I think it's really important for our listeners to understand it's not simply repackaging, but it's making some fundamental changes to the course itself that makes it appeal to a broader audience and female students in particular.

Dan: Exactly that, if you just change the name but don't change it you're just whitewashing it and that's really not doing the right service to the experience of the students. So it was really interesting, because that meeting I mentioned, that Jan Cooney led, began the creation of a new College Board course called Advanced Placement Computer Science (AP CS) Principles, and I've been involved in that in the development committee. So this is a high school course that's now in many, many high schools across the country. It's the fastest growing AP course in history. So that course really came out of that meeting, so that conversation that happened at that summit really informed a little bit what we were doing and informed obviously the National College Board Initiative.

So that CS principles course, and by the way we became one of the first five national pilots for that course, so the national pilot was a university course that says "We can teach this course at the university level, and we can also have it work in high school." Because really there wasn't a non-majors course that had those elements. If there really was a non-majors course at any university it was kind of about fluency and learning what the terms are, maybe poking at it like a fish bowl where you're not really doing it, you're learning what they are, but you're not doing it. Or it's like we were doing, as I said incorrectly, we were just having the CS1 course, the intro course to majors, be a little slower. And that can obviously help the students do well in CS1 because they've done it slower first and then you come in to full speed. But it's like a horse with blinders on it, it really wasn't opening the vision for what computer science can be and what computer science is today.

So I need to give credit to a lot of people as I talk about how we built the course. First I need to give credit to my colleague Brian Harvey, and Tiffany Barns also became part of our early team in “Beauty and Joy of Computing.” So I need to give credit to these amazing educators who have really been fundamental in making this course succeed. AP CS Principles also had a whole team of folks who built that course, so there are seven big ideas within that course, and those big ideas became a fundamental piece of our course. So those big ideas are creativity, abstraction, data and information, algorithms, programming, the internet and global impact. And you'll notice that programming is 1/7th of those, so it isn't a programming course. That's one of the things we had to sell to educators, this is not a programming course done differently, this is a fundamentally different course. So I really want to give credit to all the folks in the CS Principles team, which is a lot of high school teachers and a lot of university folks.

But our particular flavor of CS Principles, in the “Beauty and Joy of Computing”, we did some things that were a little different, above and beyond I would say, the CS Principles course. So we take a functional course to programming, so we wanted to bring the spirit of scheme with us. So that means you're not sitting in four loops, but you're thinking in really powerful abstractions. So we actually teach an idea that is not taught, as far as I know, in almost any of the CS Principles course or non-majors course or even courses that are taught at a CS1 or CS2 university level, which is functions as data. That's a natural, that a function, like 'sin of x' or something can be something you pass into another function and do that. And that allows you to not deal in four loops and deal with the low level, but fly with the birds rather than just wallow with the pigs.

We also teach a very difficult idea in addition to abstraction of recursion. And recursion is something that many people say "Whoa, don't teach that to non-majors." But it's so powerful, especially if taught with a visual language, and especially if you teach it where you're letting them create fractals and create design patterns and then make their own wonderful pictures and kind of celebrating that, rather than competing to do that, you're celebrating that. I also need to give credit to the folks who've been working with us through the National Science Foundation, EDC the Education Development Center, they're the folks who took our university course, as I mentioned we work with 400 teachers and there are 840 people on our mailing list. So almost a thousand folks are working with us nationwide and actually worldwide. There are folks in other countries doing “Beauty and Joy of Computing” as well.

They've taken our course and “high-schoolified” it, they have really added differential learning, upped their exercises more, staggered sets of that, more scaffolding, so they're a delightful design team, and it's like murderers row in terms of being able to work with these amazing folks and team to be able to bring that to high school. So they're the ones who are creating the course which is now taught nationwide in high school.

We still have the Berkeley version which is a little different, but we're kind of going to be merging it pretty soon, but essentially that's a great group. And I should also mention that we're part of the National CS For All effort, so this movement to bring computer science to K-12 and in our particular instance to the juniors and seniors and sophomores of the world. This is part of the National CS For All effort and I'm delighted to be part of that as well.

Donna: Now this sounds very different than many traditional computer science courses, and I want to go back to one of the things you mentioned earlier in the initial pilot which is paired programming. **Can you explain to our listeners what that is and why it's a part of the course?**

Dan: Sure, paired programming was not our invention. Laurie Williams of North Carolina State University (NCSU) was one of the initial leads of that, and the idea is you're never alone. One of the hardest things about being in an intro course is you get there, you're in a lab, you're asked to do a challenge and if you can't do it, you feel frustrated and you say to yourself, "I don't deserve to be here." But if you had somebody else who was with you and they're having trouble to you feel "At least it's not me." So paired programming is the idea that you sit side-by-side with another person in your class. One computer, one keyboard, one mouse. It's not two computers side-by-side, it is one computer. Which actually helps for a lot of folks who are challenged in terms of the number computers that are working in their laptop cart or something because you only need half of them.

So you have this system, you have hopefully staff helping people and circling the room, and because working with teams you're only dealing with half of those people, so it even helps in terms of the ratio of instructors to students as I mentioned. So you've got people working side-by-side, so what'll happen is there'll be one driver and one navigator. The driver is the person on the keyboard, on the mouse, with the screen. Both people have the screen, but one person's kind of doing the entering. The navigator is helping them do that, the navigator, as you would imagine in a driving situation, is helping them, "Oh, use this function!" They're looking things up, they're pulling the manual out, they're doing the thinking in some sense and the driver's just kind of doing the driving. And then 20 minutes later they swap, and the thing you can never do is you can never have the navigator grab the wheel, you can never have the navigator grab the keyboard or mouse. They can point at the screen, but they're really supposed to just be driving the process.

In fact, the navigator has the harder job. The driver just kind of operates the keyboard and types. And then you switch it, so every 20 or 30 minutes you do the swapping. And so by doing these labs, and we weren't doing this before, by doing that in labs, everyone gets to meet each other. And by the way you rotate, you don't have the same team every week or every lab, you do some rotation until people feel comfortable and find someone they really like working with and then they can lock in so you don't force the rotation on them. So there's a couple of best practices that are kind of pluses on that. But the idea is you're meeting the rest of the class, the class is getting a little smaller, you don't feel you're so isolated, and this really does help students who get to college and are freshman and don't know anybody else. All of a sudden boom, you've got a buddy. And then next week you've got another buddy. You start to feel like, "Hey, I'm part of this community." You're part of a learning community that's moving forward to get the material.

Donna: Now we have in our proven practices collection on our website, IWITTS.org, a study on peer programming done at UC Santa Cruz that had over 500 participants and a control group. And the students that received paired programming chose CS as a major and persisted at a hugely higher rate. Female students in particular, because all the research shows they prefer collaboration, but even male students also ended up persisting and pursuing CS as a major at a higher rate because of the paired programming that you're describing. Such an important principle. **Something else that you talked about that I'd like to ask you to expand on a little more is, can you also just talk about the project-based learning and the active learning that is part of the course?**

Dan: Sure. One of the things that we did is make sure, and this was delightful, we've always allowed students to choose their own projects. And that goes counter to a big movement that I honor, which is in a lot of CS1 and CS2 courses, you pick the nifty assignments, you pick the best assignments. And then rather than have a boring assignment for everybody, you pick a better assignment for everybody, and these assignments are obviously more nifty. But I really want to run counter to that, because I believe the hard thing to do is to find a way to have a fair rubric across any set of projects. Let the students choose their own projects. So sometimes I'm anti-nifty in a sense. Nifty means you pick a great thing for everybody, and I'm saying, "No, let everyone choose what's important to them." That is really critical in letting their interests bubble forward and letting them have agency.

So the idea is, students pick projects that they've always wanted to write rather than being thrown into, "Well, I'm writing a Pac Man or a guessing game." Or some kind of a data simulation. No, it's about what I care about, and if I care about a tutorial for students to learn science, one of our highlights is that students wrote tutorials for their younger brothers or sisters to do that, we can do that as well. So it allows that. And the hard part is having a fair rubric for everybody, but there are ways to get around it. If you make the rubric clear to students in advance, here's where the points are going to be generated, and just make sure they hit all these marks, we can do that. And that didn't come from us, we were doing it independently but then when CS Principles came along, I remember a phone call from Jan, I was in Chicago. She said, "Dan, what do you think about having a performance task where students will do a project in the class and have that count toward their final College Board grade?" And I said, "I love it!" Literally, they said "Shh!" Because I had taken the phone call outside of a meeting, I said, "No, this is amazing! We're already doing it and it's working great!"

In addition, we do this for programming, and I've talked about only programming so far, but one of the really important things I need to talk about is the social implications of computing. As I mentioned one of the seven big ideas was global impact. So that's one of the things that we were never doing in our early course but we were so delighted to bring in some of the things that Brian Harvey had been doing. And he's in our seniors and juniors version of social implications in computing course that I'm now currently teaching, we're bringing that down into the non-majors classes. So every single class we bring a piece from the news, and we say, "This is how computing is affecting your life and my life." And what I've heard from some high school teachers, this is in terms of best practices we've learned from our PD, professional development, is let the students choose the news items.

So now you're letting the students go and pick two or three, they'll read the *New York Times*, they'll read CNN, they'll read *Technology Review*, they'll find some news that they care about. Maybe it's more about video games than not, whatever, they care about it. And they're bringing that and they're sharing it with the class. So we've heard that is the best practice for folks in high school. Again, there's that agency, what do they care about? They care about loss of privacy, they might care about Facebook, they don't care about the corporate issues. Mostly when you read the tech articles it's mostly about the business, "Oh, Amazon stock is doing this." They don't care about that as much, they do care about the apps that they're using and the privacy that they're losing. So the students bring that and that really, really does work.

So bringing computing in the news every day to your classes. And that doesn't have to be something about just non-majors classes. That could happen at every single computer science class that's happening across the country at every level. Here's what's happening in the news, here's how it's affecting you.

Donna: So I'm curious, do you see some differences when you have students choosing their own projects? Do you see a maybe more diverse set of projects than maybe what is given out across the board as standard projects?

Dan: Yeah, it's interesting you mention that. There's the obvious thing, which I'm sorry to my students who have fallen into the trap, but we don't let them have violent games. If it's a game it can't have any shooting aspect to it and no gore obviously, we don't want to have an 'R rated' game. But we still have, you know, you're a rocket and you want to shoot down the thing. That's okay because it's not gory and you're just doing kind of a Pac Man era thing. I see men choose, my male students, choosing the shooting games more. There's a tank game, there's kind of a zombie thing. Whenever you have games the games end up being kind of Mortal Kombat-y things. But they always tame them down, but it's more kind of battle games from men and I almost never see that for women. What I do see from my young women students who choose a game, is in the same way, you remember Ms. Pac Man was the game that many women at the time were gravitating towards. So I'll see Pong and Snake and Connect 4 and other board games and traditional games. I don't typically see violent video games. So violence does seem to bubble its way down across gender lines, so I see less of that.

I also see more tutorials, a lot of female students choosing things to teach other students that have more meaning, to bring more meaning to the world. When we were picking a couple of projects to choose to highlight for our edX class, we offer a class on edX, an online learning class. And we didn't do this intentionally, we just picked a couple of star students who had won some awards. Every December we pick the best five projects and we have them stand up in front of a group of 500 local high school students and then we show them projects. We don't pick them to kind of make diverse projects, we just pick the best five and we hope that they're diverse. And the ones we picked were tutorial about teaching science and the one that we picked from the other team that was chosen had two boys and it was again a tower defense game, like you drop some towers down. So it really was not intentional to do it across those two lines but that was exactly a microcosm of the kind of projects that women and male students choose.

Donna: Great. And I, myself, was asked, "Should we do away with all of the games in computer science?" on an NPR interview, and I said, "I don't think we should do away with games, I think we should expand it so that it also includes social interests or tutorials or things that help, so that you have the full range of things that will engage students." But I really love what you've pointed out to our listeners, which is that when the students themselves get to choose the project, I think that that helps with diversity of projects, but also it's what most engages them at all levels. So I think that's a really important principle.

Another thing that I want to ask you about, before I move on to some of our questions, is can you just talk about in terms of active learning in the labs? I know that in our pre-discussion before you said that is also a very important element of your course, so could you speak to that a bit as well?

Dan: Sure. So the active learning actually happens across the different elements of our class. There was a movement that I really believe in which is "moving from a sage on the stage to a guide on the side" and what that means is you're moving a lecture-centric course to a lab-centric course. And my colleague was really a proponent of this and a huge influence on our design. We have seven contacts with our students a week, and that's the most on the campus. We said, "As we're making a new course, let's get the most time with our students, so when they're getting confused they're with us." So we changed it from one hour, two hours of lab a week to four hours of lab a week, and that's a big deal. It means you have to come up with four hours of activities to do.

This happens in high school, in high school this isn't unusual. People know you shouldn't be talking at them. But that hasn't bubbled to the university, people are still creating courses today where they're basing it around the fact that they are the sage, they want to make a course where they're just going to talk on stage a lot. That's not where deep learning happens, deep learning and real investment happens when they work on a project they care about. So that's critical, is having it lab-centric. And as I mentioned, it's paired programming so they're really enjoying their time in lab.

Even if you have some lectures, we do have two hours of lectures a week, we have it active learning. So the active learning angle is that we have clickers in our class, and that means roughly I'll go 10 minutes or 15 minutes or sometimes even five, and then ask a question and pause, and then we give clickers out. We've actually put some investment in so the clickers are actually given to the students--they don't have to pay for them. So the entire course is free, that's unheard of, to give clickers out on our campus. But we get the clickers out, we get them checked back at the end of the year.

So you ask a question, the students think about it, mull it over, and then answer. Then you see the histogram, and then then you let the students talk to each other. This is called peer instruction, this came from Eric Masseur of Harvard University who is a physics teacher who wrote a book about it. So the idea is you talk to your neighbor about it, you're teaching your neighbor why you thought it was B and they're teaching why they thought it was C. And then you decide to hold hands and sing Kumbaya and you have to vote the same vote. So you have to kind of convince each other and come to a consensus. And then as the instructor you take a look at the histogram and you say, "Oh wow, you're all perfect, you're all on task and you all got it right, so let's not dwell on it and move on." Or if they're not right, you can stop the class right there. So you have to allow a little bit of flex time in class to allow that active learning to happen. As I mentioned, the two big things is we move from sage on the stage to guide on the side and we have active learning and lecture.

Donna: And what I love about the clickers is that there's a lot of research that shows that the majority of female students are reluctant to ask a question in front of the rest of the class, especially in a large setting. So when they're using the clickers to give feedback they don't have to ask a question in front of everybody and they're still part of the feedback loop. **The other thing, in terms of talking about your lab hours, could you talk about the lab assistants? Are they diverse?**

Dan: Oh my goodness, that I think is the thing that really is the hallmark of our class. From the beginning, it wasn't like this course invented the lab assistant model, they're called academic field study, folks who are just working in an environment where they're getting credit to help. And you can do lots of different things for academic field study, you can do outreach in high school. But in our version of academic field study they work in and support the labs. So these are students who get credit to help with this, and we're mentoring them. It's like a class, so we're helping them be better lab assistants and better teachers. And that mentoring really makes a difference because the way you work into our student staff is you become a lab assistant or you are an academic field study assistant. And then you move into a reader, which is a grader. And then you move into a teaching assistant (TA), and then you become the head TA, and then we actually have opportunities for summer instruction.

So you can be the lead instructor if you work your way all the way to the top, and it's amazing to watch how good they are, and it's hard to pick the best "lab assistant" to become the grader, harder to pick the best grader to become TA. But we do that and we get great, outstanding folks. But when we looked at it, even in the beginning, as I mentioned our class was 45% women, so if you just take a scoop of these amazing folks who really are interested in teaching and have them be lab assistants, you're going to find 45%. So we regularly have 50% or more lab assistants, and that has influenced our readers, who are now 50% female, has influenced our TAs, and now I have 12 TAs and guess what, six are female and six are male.

When you look up, you're in lab, or you're down deep in code, and you look up and somebody who's coming to answer your question looks like you. And that's not just women. Also represented are minorities who are part of our group, we haven't mentioned that yet, but we're doing great with that. We're setting some campus records for that in terms of intro courses. But we haven't been as public about that because we're still working on those numbers and we'd like to be better. But the key is, all of that brings into the ecosystem of the course, where everywhere you look it feels gender equal. And right now I have two amazing head teaching assistants, Laura and Monsie, and guess what, they're two wonderful female students. So the leadership of the class is female driven, it's just delightful at all levels to have gender equality and not have to have it be a big deal. It's just that what happens, if you have a pool of students and the pool is diverse then the staff that you're drawing from is diverse.

It really feels that once you get over the hump, once you get over the past point where you're kind of around 50%, it's not a big deal. It's now going to be this way forever because I can't imagine anything changing about the system of CS10. So that's a key piece of that as well. We haven't mentioned this, but also it has to be mentioned, it's a graphical language we use which is now Snap!, so BYOB became Snap!, which again is led by Jens and has some support from SAT and other folks. Snap! is amazing, it's pretty, it's a pretty programming language, and you don't have the textual syntax errors. So people are still living at the intro space still thinking that the best way to teach young kids to program is typing, and people can't type. They're making comment errors, they're forgetting a semicolon. They're just frustrated by syntax. And if you live in a blocks world it really does look like pseudocode. So there are just so many benefits at the intro level. And there's a couple of conferences about that and talking about the papers proving that.

People need to kind of realize and wake up to the fact that blocks programming is the way that you should be teaching intro programming, and many people have kind of bought into that, but many people haven't. What's nice is that some systems actually allow you to go back and forth, which is really nice. So the first “N” weeks you're in blocks work, and then in the same system you can drag a slider and it becomes text. There are some things you can certainly do better in text than you can blocks, like copy and paste and replace all, and many, many things at the early level that you can do better at a block level. So we stay in Snap! in the blocks language, and then at the last part of our course, because our next course for majors is in Python, we actually have a smooth transition to Python so that when they get into the first course for majors they've seen Python before.

So again, one of the goals of our course is to serve the campus, one of our goals is to, of course, is to provide them with success in their first course for majors. So by having kind of a unit in Python, it really does help flow into the first course for majors which is really nice.

Donna: One of the things that I'm hearing, and I just want to comment on, is that in the those initial two years you did the white boarding and the brainstorming and you've given credit to many other educators and leaders at the National Science Foundation at the program level to come together and develop this. But once you did and you piloted it and it was half female and you incorporated all of these principles that we've talked about, it sounds like it's been pretty organic. So you've got what I would describe as a positive loop on broadening participation, for women in particular but also, as you say, for minorities, people of color, and that's something that you're still working and growing on. So it attracts female students, then they become part of your teaching, labs to teaching assistants, and so then you've got this organic loop where you broaden participation, which is really a beautiful thing.

As a last question, before we go on to the questions that we have from participants, for those who are listening and would like to get started on incorporating some of these principles or creating their own organic broadening participation loop in their CS course, how would you recommend they get started?

Dan: There's a couple things, I've been thinking about this a fair bit. If you have an existing course, let's say you're one of a team of folks that has to teach a course and you don't really have the opportunity to revisit and do as I said, put the whole thing on the table and whiteboard the future. If you can't really redesign your course, here are lots of things that you can do already in your course, in every computer science course across the country. Number one, let the students choose their own projects. We mentioned the importance of that. Number two, think about a paired programming approach to how you do your labs if you can. Number three, think about computing in the news. Think about bringing the impact of computing, the social implications of computing. And don't just use computing in the news to talk about the latest feature, the new hot things. Talk about some of the issues, talk about some of the privacy implications, talk about some of the data breaches, talk about some of the problems with Cambridge Analytica. Think about those things.

Bring active learning to your lecturing. If you're going to be doing lecturing, try to stop every 10 minutes and have these clickers. There are lots of other pieces of software that you can do, there's clickers, which is you print something on paper and it has different patterns. And the cellphones, you can hold your cellphone and the cellphone can read the room. It's a really delightful non-computer technology that can do the same kind of thing. So basically break up your lecture, if you have any lectures at all break that up and do that. What I didn't mention about this is if you have guest speakers, try to make your guest speakers diverse. In “Beauty and Joy of Computing” this semester we've got six guest speakers and five of them are women. I leaned it that way because I'm on stage for most of the class and I'm bringing a Hispanic male voice to the table, but let's bring some diverse female voices to the table.

So I'm having some of the best female professors, who are doing some exciting research and they're amazing. And they're both allowing a different voice to be heard, but also they're letting young women say, "Wow, I could be that. Look at that, that's someone I could be." There's a common phrase, “You have to see it to be it.” So if all you see are the traditional white bearded men teaching you, then you don't really see yourself in that position because look who's the professor now, but having a more diverse faculty certainly helps. If you can't make changes to that, that's obviously a slow process, bring guest speakers that are diverse as much as you can.

Think about project-based learning, think about less lecturing and more lab and project work and putting more emphasis on that. Think about project work in teams, so not just a solo project but allowing teams. And a lot of people say, "Well, if we let teams happen then you'll never have the proof that the student knows their material." Well, that'll happen through the exams and that'll happen through other courses, but in the intro courses certainly let them work in teams up to two or three. We haven't tried bigger, although bigger has been attempted by other folks. And again, in any way possible think how you can connect your courses to this interest.

One of the things we did, it's fascinating, we always had an essay. You're not just reading and writing about this data privacy. By the way we have lots of links and reading assignments. Our “Beauty and Joy of Computing” course has reading assignments every week. You've never heard of a computer science class having a reading assignment. Sometimes it’s a TED talk, sometimes it's a New York Times report, sometimes it's something deeper. But we'll try to have that element, and we try to back that up with having reading quizzes once a week to make sure that they do that and we have discussion time to talk about that as well. But one of the things my TA Luke Siegers did, he came and said, "Dan, you're having students write an essay, a page or two of what they've learned or of something that they want to research, and you only hand it to us the staff who grades it and gives it back to the students. Well, why don't you let the students grade each other and give comments to each other?" So I said, "What a delightful idea!"

So we don't let them grade each other fully, but what we let them do is we have them, we force them, to post their essays in a public forum. We use Piazza, a delightful website we use to kind of allow student to student teaching happen. So we create a special Piazza for this and we have a special forum for this and we have the students post their essay into the forum, and then we require that the students comment on each other's essay. So again, it's not just of the agency of "Well, I got to write about what I care about," but I also got to read about what I care about. We force the students to read the student the left, to read the student to the right, and then third option is they read any student they want to.

So you have a student who writes something particular interesting about the new video game or virtual reality, and all the students who have that third free, they can read any essay, they'll read that student. So you kind of get a sense of how popular your essay was by how many students choose their free essay to be the one that they read. And you have to read it, digest it, and just like you'd have on a LinkedIn post, someone posts a thing and you might write a paragraph response, we ask everybody to digest it and have a paragraph response. And we grade them on the paragraph response, we grade them on, "Did you really the read the article the person read, and did you have a thoughtful response to that?" So we create this kind of dynamic social aspect to the social implications of computing which has become a really wonderful thing that just runs itself, as I mentioned.

So those are a couple of things that you can bring into almost any course. Everything I mentioned before is things you can do at any course. At a larger scale, if you're teaching a non-majors course, really do think about CS Principles, think about some of the delightful work that's been done--us and other courses in terms of copying our work. Most of the folks who I know, who created these non-major courses, have their work available as Creative Commons. Which means everything that I've done is available for free to anyone, all you gotta do is give credit, that's all you gotta do. It's yours. And don't make money on it. It's noncommercial, ShareAlike, take it and use it. We even have a modification lab, so you take it, you want to twist it, you don't even have to ask me. The Creative Commons license we've chosen allows you to just borrow whatever pieces you want and it's yours, just make sure you give us credit at the end of the day and don't earn money on it, which is fine.

So most people are doing that. So grab the best pieces of these courses and build your own non-majors course and really rethink that, bring in data that students really want to use, and let the students use their own data. So as much as possible try to rethink your courses and rethink what's really important. It's always healthy to rethink your courses for the new age, but as much as possible think about what you can. And take some of those seven great ideas from CS Principles and bring those into your course if possible.

Donna: Those are lots of great suggestions. So as you say, even if you yourself are an individual professor or instructor, you'll have many avenues for getting started. I myself attended a couple of your classes online in preparation for this interview, it was a lot of fun. Now it's time to take the questions from the listeners. I see that we have a number of questions already.

So the first question is ***"One of my favorite first courses in computer science taught us to solve problems and allowed us to choose a language to use since languages have various strengths and weaknesses. What do you think about this approach?"***

Dan: That's a fascinating question, thank you for that. That's even taking my "let the students choose" one level above that. I would say that if you have never programmed before, I don't know if you would know where to go. I don't know if you'd have a place where, "Here's 12 languages, choose one of them."

And then you've got kind of the Tower of Babel where everyone chooses a different one. So I can see how that could work for maybe a little bit later on in the process. There actually are different approaches. I did a study with Jeff Forbes of Duke University, we interviewed the top 32 PhD-granting institutions as chosen by *US News & World Report*, and we asked them what they were doing in their first course, second course, and third course for majors. And every once in a while someone would take a very language-breadth approach.

What they were doing is in their intro course they said, "You know what, there's a ton of languages out there, we're going to expose you explicitly to six or seven or ten of them, and talk about the various benefits of each one of them." So that's kind of a different approach on to the CS1 space, and again it's controlled by the instructor, they have exercises in that particular language, and they're kind of doing it lead from the team. I think if you do it wide open, I think it will be harder for students to know what to choose in terms of what the benefits are. They don't even know how to do a Hello, World! yet, so you've got a language. The other challenges might be that the teaching staff might not know the language, and so when a question comes up if you actually have no formal language, other than a stack overflow or something, if you have no way to give those resources to the students that can really be a challenge.

That said, I'm volunteering in high schools, Technology Education and Literacy in Schools (TEALS) is an amazing program that comes out of Microsoft, and they let anybody volunteer in a high school as kind of a co-teacher or lab assistant model. So I'm volunteering at a high school for a year, teaching BJC. It's kind of a funny "eat your own dog food" experience. And the teacher I work with, Ms. Jen Dirkson, is amazing. And one of the things she had done the year before I got there was she taught the intro course, and then the last month or two she lets the students do any project they want to, including teach themselves a new language and then do a project in that language. So a student wanted to do some robot stuff and they learned how to do this. They wanted to do some Arduino, they did that.

So there is a space to do wide open projects. In fact, for our final projects, we let our final projects be in any language. Now most people coming into our non-majors course know no other language and so they've only seen Python or Snap!. We certainly encourage people to do Python and Snap! because we'll support that, but if someone were to do something else, I don't know what I would say in my own course if they chose one of the two languages we haven't chosen, but Ms. Dirkson, she lets them choose anything. Even if she doesn't know how to do it, she'll work with them, "Well let's find resources on the web!" And she works with them. So I really do appreciate that kind of letting them be open at the end for a project, and even if that being open at the end includes letting them choose their own language, it's quite interesting. So I think there's ways to do that successfully, I think it'd be a challenge at the starting point for CS0.

Donna: Sure, sure. Well we have two questions related to pair programming next. So one of them is ***"Do you explicitly train students to be supportive partners when they're doing the paired programming?"***

Dan: So we do that in a couple of ways. One of them is, there are a couple of videos online about how to do paired programming well and how to do paired programming poorly. It's kind of a funny video where a pair pretends to be really a bad pair, where you're grabbing the keyboard. So we have students watch that. We also in our early labs actually, we kind of roughly timed how long each page takes, and so kind of on the fifth page, which is about 20 minutes, by the way don't tell the students that, it will say, "Okay, now time to switch." And there's a little animation that shows the partners switching seats. So we do this for the first couple of labs that reminds them, because otherwise they'll forget and they'll just lock in as one person's driver and one person's navigator for the whole time. And it's really easy to get into that, if someone is a faster typer and you feel better in your role, it's really easy to get stuck in that rut where you don't swap.

Also by the way, our TAs are way trained in terms of what makes a good paired programming and what isn't, and when they see it not being so healthy they'll intervene. So I think it helps for both the staff to know what makes a good team and what doesn't to be able to say, "I just saw you grab the keyboard and reach across, not so good." Or "I see you're not tilting the screen so the other partner can't see." They might have one superstar and one weaker student and it might be how the pairs happen to be chosen and you'll see the superstar just keeps the screen faced towards them and really is just ignoring, so you gotta step in and make sure that doesn't happen.

There's also been work in terms of how do you choose those pairs? Do you let the pairs randomly choose each other? Do you ever let the top student in your class pair with the weakest student in your class? So that can be a challenge if the difference is really far, so trying to keep the rough differences in abilities roughly the same can also be something you can do help successful pairs.

Donna: And that was the other question, which is ***"how do you suggest pairing students? And so do you in fact pair them versus letting them choose themselves?"***

Dan: So what we typically do in our class, and there's lots of different ways to do it and people are locked in on their way of being better. We let them choose their own pair, but then we force a rotation, and I think I mentioned that. So it's not like you're locked in a pair, because if a pair is dysfunctional you want to get out of that. What's hard is if you have an odd number of people, and if you have an odd number let a pair of three happen. And that's a little bit harder, to have all three be engaged, but at the very least nobody has to work solo.

So you let them pair randomly. I'm not in the labs as much, my TAs are in the lab, but I guess sometimes if someone doesn't want to work with somebody, it's like you get picked last on the basketball team right and it's obvious that you were not the one, so if you have trouble finding a partner that can be I guess hard. But finally, you get somebody and the TAs can help with that. One of the things we do to make sure there's always a balanced team, you line everybody up and kind of do some shuffling like a random seating then and just sit down quickly say when the music stops. You don't have to be as silly as that, but if there's any issue of how to do pairing that can work. But then you tell them they can't ever pair with the same person again until I tell you. The next "N", well the question is what is "N"? How long do you force these rotations? Do you make sure everyone's paired with everybody else before you let them choose their favorite person to work with or not?

And what's actually funny about it is sometimes when people choose their best friend, they're not the best team because they talk too much. That was how it worked in high school, and we had people who were always best buddies and they end up being fun and high-fiving, but they're actually not as efficient as if they worked with somebody who is more serious. It's actually kind of funny when you let them choose their favorite person and that's not the best for them.

So then what we do is we let them go five or six labs in of forced randomness every single lab, which is a two-hour lab. And then the seventh, and by the way just by the seventh, that's when they need to start grabbing a partner for their projects. We actually force project work as well, we don't let them work solo for projects. So just about then it's time to start thinking about your first project partner. So you're kind of settling in your lab as who's your person you'd like to work with.

It's funny, I haven't been there for that, but that's kind of like getting dates for the dance. Who chooses what. Obviously there's uncomfortable portions. Maybe there are better ways to run that, but where you're not being chosen, you finally get chosen, and I guess that's the person you're working with. So I haven't been there for that, but I guess there's always that uncomfortable last stage of how that's done.

I guess it can be done well, this also is what's called the stable marriage problem. Where everyone could put in their preferences and you could run software that says "I put in all your preferences of who you want to work with!" And then we've chosen based on optimizing your preferences, so that's often how school choice is done at some of these lottery schools and some of that. So you could do it with more computer science but we just kind of let it happen organically, but again we could do it with a little bit more heavyweight pieces of software behind it. We haven't done that though.

Donna: I went to a presentation with someone who was a physics major at Harvard and his mother was a diversity officer at one of the Ivy League colleges, and he actually created a piece of software that did exactly that, like he put in all of the weighting factors and et cetera. So it's interesting.

Dan: You could rank order your top three, "I've worked with seven people, here's my top three." And then try to optimize that.

Donna: Yeah. So two more questions. So one of the questions is, ***"Could you see your type of course at a community college?"*** We have a lot of community college listeners.

Dan: Oh my gosh, 100% yes. You had me at hello for that one. This is a non-majors course. There are people who are coming in from high school having never taken computer science before, so it's great for this course, CS Principles is great for that. I got a four-year school and there are people coming to our four-year school who haven't programmed before. There's a great space for that. So in terms of kind of modeling for that, you obviously have people coming to community colleges who haven't programmed before. And even if they haven't programmed before they might have done a little bit, and it's really nice to have a great course like that again, like CS Principles like “Beauty and Joy of Computing”, really does capture much more than just programming.

A lot of times I see more vocational courses at two-year schools, because you're trying to prepare people for the workforce, and that's a very reasonable thing to do, but you also would like to have a course that summarizes it all. This computer science, this is the field. This is the limitations of computing. One of the lectures that we teach is the “Limitations of Computing”. Computers can't do everything. There are limits to size of the numbers you have, limits to what kind of problems can be solved, and how long it takes. So those things are what makes this course special, so there's certainly 100% room for a non-majors really inspiring course at the community college at two-year levels, certainly.

Donna: Great. Now I see a question, ***"What would be the best URL to go to see your videos, for those that are interested?"***

Dan: Sure, so the main place, this is the single link we give to everybody, which is https://bjc.berkeley.edu/. So hit that website, and when you go there you'll be able to both see our professional development opportunities, I should give a plug for that, we offer opportunities for high school teachers across the country, and for community college folks and for university folks to come and attend our PD sessions. So this is usually a week-long session offered all across the country to learn about our course. As well as the BJC PD opportunities, you can see our curriculum, so there are links to the Education Development Center (EDC) version to our curriculum which is kind of now the high school version of that. There are links to our Berkeley course if you want to see how we're doing it at Cal as well, and you can ask a question, contact@BJC.Berkeley.edu is the email, so ask us if you want to kind of join our piece.

There are links to our edX course, which allow for not just the videos, I want to say that you don't just go to the edX course for videos, you go to the edX course for videos and auto grading exercises. I have some outstanding students, Michael Ball and Lauren Mock have worked very, very hard to make sure that our autograding works really well. So we've got autograding. It's actually very unusual for a blocks-based programming language to have autograding. Normally you have a text language, you submit the piece of code, and you get back the results of how your test passed. So this is where, you're doing this in the website, so people have worked really hard to make it kind of a full experience here at edX. You're watching a video, you're having a clicker question that asks you a question about it, then it says "Oh do this challenge" and here's the description of the challenge. And right in the window is the Snap! environment embedded into the JavaScript window. And then when you click it, you say submit, and it will tell you if you passed the test that we had.

The tests aren't just unit tests where we're making sure that the functions you write are correct. We're even looking into your code and telling you, "The way you did it isn't the way that I would've done it." So we've actually added some introspection into the way that the autograding exercises work. So all of that is available on edX.org and that's also available on that website.

Finally, if you're a teacher and you want to use our edX course as a small private online course, you can grab the course and use it with your small course in your small classroom. So if you've got 20 students, you can use this edX online resource really kind of meant for the masses and hundreds of thousands of students, you could use it as kind of an eBook in some sense. Mark Guzdial and Barbara Ericson are from Georgia Tech, now at Michigan, they've talked about having eBooks be the thing. So it's almost like using this MOOC as an eBook, and that idea came from Armando Fox, my colleague. So the idea is that high school teachers are using this research as "that's the lectures and that's the labs" and it drives it all.

So if you want to go to just our BJC material you can see it and the labs are there, but if you want to kind of have the videos as well you can go to the edX Small Private Online Course (SPOC) world of it. So it's kind of two flavors you can do if you're part of our BJC family, so all of those resources are at BJC.Berkelely.edu.

Donna: And I love that they're available, with of course giving credit, and also to modify. That's wonderful. Well I want to thank you Dan, so much, for being our second opening speaker of the Telsummit. I really enjoyed our conversation.

Dan: My pleasure. Delightful to be here.