Transcript

[00:00] [background music]

Derek Bruff: [00:10] Welcome to "Leading Lines," a podcast from Vanderbilt University. I'm your host, Derek Bruff, the Director of the Vanderbilt Center for Teaching.

[00:13] In this podcast, we explore creative, intentional, and effective uses of technology to enhance student learning, uses that we hope point the way to the future of educational technology in college and university settings.

[00:25] In this episode, we talk with Akos Ledeczi, Professor of Computer Engineering and a Senior Research Scientist at the Institute for Software Integrated Systems here at Vanderbilt University.

[00:37] Akos is the lead developer for NetsBlox, a graphical programming language designed to introduce novice programmers from middle school to college to networked programming.

[00:46] Students can use NetsBlox to create simple multiplayer games and to build apps that interface with publicly available data sets. Akos is interviewed by Cliff Anderson, Associate University Librarian for Research and Learning and a member of our Leading Lines team.

[01:01] Cliff and Akos discuss the past, present and future of NetsBlox and explore how graphical programming languages like NetsBlox, Snap and Scratch are changing computer science education.

[01:13] One terminology note, Akos uses the term blocks-based coding to refer to programming languages like NetsBlox and Scratch, in which graphical interfaces allow users to drag blocks of instructions together to create relatively complex programs.

[01:27] [background music]

Cliff Anderson: [01:27] Hi, this is Cliff Anderson. I am the Associate University Librarian for Research and Learning and Professor of Religious Studies at Vanderbilt. I'm here with Akos Ledeczi, Professor of Computer Engineering, Director of Graduate Studies in Computer Science and also Senior Research Scientist at the Institute for Software Integrated Systems at Vanderbilt.

[01:50] Welcome, Akos.

Akos Ledeczi: [01:51] Thank you.

Cliff: [01:52] Today, we're going to be talking broadly about the area of computational thinking, but in particular about a new tool that Akos and his team had put together, that is called NetsBlox (https://netsblox.org/).

[02:04] One of the things, before we get into what NetsBlox is, is to talk about how you became interested in this area of blocks-based programming and model-based computer systems.

Akos: [02:18] Yes, thank you. I took a really long and roundabout way of getting where I am right now. When I started with my graduate studies in the early '90s, our main research area was this model-based engineering or what we call model-integrated computing.

[02:35] The idea was that there was very little modeling done in software engineering at the time and then computer science, so we tried to bring in a more disciplined approach to creating software systems.

[02:46] The idea was to create domain-specific modeling languages. These are graphical modeling languages to software engineering. The reason to do domain-specific and not one universal modeling language because software is being used in all kinds of different domains and to us, it never made sense to just try to create one modeling language to capture all.

[03:08] In different engineering disciplines and different science disciplines, there are all of the existing modeling formalisms, so why not create an environment and the modeling language that fits that domain well.

[03:22] That was our idea, and we created a series of tools where you can actually define a modeling language also through graphical means. This environment would configure itself to support that domain-specific modeling language.

[03:36] There are actually existing examples of domain-specific design environments already being widely used. One example would be LabVIEW (http://www.ni.com/en-us/shop /labview.html) or Simulink Stateflow (https://www.mathworks.com/products/stateflow.html) from MathWorks. These are graphical languages helping engineers create signal processing systems, control algorithms or laboratory experiments.

[03:58] Of course, if you don't have a big enough market, it will be really expensive to create a tool just for a small market. Instead, we have this generic tool that was really easy to configure to create a modeling language, and through tool support and APIs, write code generators.

[04:14] Part of the system would be modeled with these graphical tools and the code would be automatically generated. We have actually many successful applications of these techniques throughout the years.

Cliff: [04:24] In a way, this connects, if I understand correctly, with your interest in blocks-based programming or graphical programming. I'm not sure which term you prefer.

Akos: [04:35] That early part of my career was really based on modeling, so we didn't try to model the software. We tried to model the system, and then generate the software from there.

[04:45] When I started teaching introductory programming with MATLAB to freshmen -- non-major freshmen actually, non-Computer Science majors -- I noticed that coming with zero programming background and learning text-based programming language was a pretty big job for many of the students.

[05:05] That's when I started to introduce one of the block-based programming languages, Scratch (https://scratch.mit.edu/), just for the first two weeks of the semester. The students actually liked it and it helped them with the initial learning curve, then we switched to MATLAB. That's how my interest in block-based programming grew and this model-based approach and the block-based came together.

Cliff: [05:31] We were talking before we got started, that graphical-based computing languages have actually been around for a while now. In fact, Logo (https://people.eecs.berkeley.edu/~bh/logo.html), which was created in 1967, when paired with Turtle graphics (https://en.wikipedia.org/wiki/Turtle_graphics), has been used for a long time as a tool to teach computational thinking in classrooms. I remember using that in the 1980s.

[05:49] What's fun is that my son is now using it when he's reading Gene Luen Yang's and Mike Holmes' series, Secret Coders (http://www.worldcat.org/title/secret-coders /oclc/930298871), that uses Logo programming to teach basic computer science concepts.

[05:59] Especially fun is the fact that at the end of the book, there's a program listing that he has to type in to get a secret code. I remember doing the exact same thing in the 1980s when I'd get magazines.

[06:09] To learn a game, I'd type in the code listing at the end of the magazine and then see how it worked, which was a great way to learn about computer programming. In any case, you mentioned Scratch. Can you talk a little bit about how you used Scratch in that course and what you found its limitations to be?

Akos: [06:27] As I said, I started teaching MATLAB (https://www.mathworks.com/products /matlab.html) and it took us, basically, a month to get to loops and then the statements, because it's more complicated, the syntax is harder. I started looking for ways of easing this transition from no background to programming. And then I found Scratch, and it was really amazing because it was so intuitive.

[06:51] It was a great way of trying to hide a syntax or not get bogged down with the syntax, but think about the algorithms, the computational thinking and not worry about what you have to type, why does the compiler complain, and stuff like that. I really liked Scratch, it was really intuitive. Basically, when I started doing it, we were able to, together, write our first program with loops and if-statements during the very first class.

[07:20] It didn't take us a month to go there. It takes us half an hour to go through some of the most important concepts in programming. That was really great, and my philosophy is to learn programming, you have to do it. If you just come to a lecture and listen to what I'm saying, you're not going to learn programming.

[07:39] So immediately the first week, I assigned them a sizable project. They had to do it in Scratch. Basically, the first week, they have to do programming, and it turned out very well.

Cliff: [07:48] Probably, there's fear when you're facing a blank page, like any writer has, when you don't even know where to start. Scratch gives you that palette and lets you drag things on and experiment. It's just much easier to begin.

Akos: [08:02] Exactly. Now that I'm teaching in various schools in Nashville — not just at Vanderbilt, but various middle schools and high schools — some of the feedback I get is exactly this. "It's so much easier to do it because I can just see what's available and start using it like Lego blocks, and don't have to start with a blank..." [laughs]

[08:24] [crosstalk]

Cliff: [08:24] If I understand correctly, your project builds on Snap (https://snap.berkeley.edu/), which is a variant of Scratch, that introduces more powerful computer science constructs. Can you talk a little bit about how Snap advances the block-based programing paradigm from Scratch?

Akos: [08:42] I think Scratch is great. The best demographic is elementary school and middle school. Once you get to high school — and of course college freshmen, like I did — some of the students would consider it too childish or too limiting and things like that. Snap is actually great because it added some important, more advanced concepts to what Scratch supports.

[09:11] For example, recursion or full-fledged functions called custom blocks and some higher order functions. There are some features of Snap that I have never used even now.

Cliff: [09:22] I see it allows you to do continuations (https://en.wikipedia.org/wiki/Continuation), which is a unique feature in Scheme (https://en.wikipedia.org/wiki/Scheme_(programming_language)). It is pretty advanced.

Akos: [09:29] Right. I think what they call it is a graphical version of Scheme. That's exactly what they tried to do. That is great. Of course, there was one huge advantage over Scratch, which is its open source and it's implemented in Java Suite. We were able to take that open source code base and add blocks on top of it.

Cliff: [09:48] Let’s come to NetsBlox. NetsBlox, as I understand, it has two major innovations. One is message passing and the other is distributive programming. Can you talk about why you added those two features in particular as your innovations on Snap?

Akos: [10:01] If you go back one decade…Two decades ago I was working with model integrated computing. A decade ago, I started working with distributive programming, specifically wireless sensor networks. I really like these two areas.

[10:19] When I started using Scratch and then Snap, I realized that even though Snap is much more advanced, it’s still limiting because it still puts you in a sandbox, which is the stage where your turtle graphics objects can move around and whatnot. You are still limited to that sandbox.

[10:38] The idea was, these days, almost any application used on your phone or your laptop is network. You go to Google Maps. You listen to Pandora. You go to Netflix, Facebook. Everything is networked. I thought that was a really big missing piece from these block-based languages to open up the Internet. How can you do that?

[11:02] We started brainstorming with my team. They came up with these two abstractions on this message passing so that actually you can pass messages between two computers.

[11:15] Our goal was basically to enable multiplayer games. Kids play multiplayer games all the time. How about making it easy for them to create multiplayer games? That should spike their interest. That was one reason we introduced message passing.

[11:30] The other main concept we introduced, what we call remote procedure calls, which allows you to call some kind of function, some kind of procedure in the remote machine. That remote machine, in our case, is our own server.

[11:44] There, we basically interfaced to a set of public domain databases or data sources on the Internet such as Google Maps, Weather, USGS Earth Screen Data, or the Sloan Digital SkyServer for Astrodome Images to open movie database and a bunch of others. Now, kids can write programs which can access these data sources.

[12:09] For example, in a five-minute video, I show how you can create an interactive weather map where you have the entire world show up as a Google Map background. You can click anywhere and it shows you the current conditions anywhere on the world. It literally takes five minutes to create that application with NetsBlox.

Cliff: [12:29] I think it's wonderful, especially that you provided so many different examples so that people can get started, sort of see how the code works and then see what the code is exactly, what the blocks are and then take it from there.

Akos: [12:37] Right. The whole thing is obviously iterative in the sense that these kinds of applications, these network applications that get data from the server start out easy just like this weather application. Of course, you can get more and more complicated when you get to the message passing.

[12:52] Again, you can do some relatively simple applications like implement a quick chat room, or text messages, or something like that. But once you get to the multiplayer games, now that's a relatively big jump because now you have to think about distributed algorithms.

[13:06] It's much more complicated than just a simple algorithm. Now two machines running in parallel has to coordinate and communicate with each other. That's where the ceiling is really raised for learning more advanced concepts.

Cliff: [13:24] Yeah, I would say that a large majority of computer programmers in the professional world have difficulty with those concepts.

Akos: [13:30] Right. Except, NetsBlox makes them a whole lot simpler than you would have to do it in Java or anything else. These are the kinds of games we support, more like either board games or games where there is some limited amount of animation.

[13:48] Of course, you're not going to implement a first-person shooter game. That's just not what it was designed for.

Cliff: [13:55] Another area that's caught the attention of the educational technology community are microcontrollers. I know there have been people developing blocks-based programming languages for controllers like the Arduino (https://www.arduino.cc/). For example, ArduBlock (https://github.com/taweili/ardublock), mBlock (http://www.mblock.cc/), or S4A (http://s4a.cat/). Have you thought about developing a version of NetsBlox to use on the Arduino?

Akos: [14:15] Yes, definitely. Actually, there is a version of Snap called Snap4Arduino (http://snap4arduino.rocks/). The developer of that version and the Snap developers are working relatively closely. By the way, we are also working with that same team closely because they like what we are doing. So yes, definitely, that's an interesting topic.

[14:38] What we actually would like to do is even better, which is we want to introduce robotics. There are of course already existing robots that you can program with these block-based languages.

[14:48] What NetsBlox can bring to the table is using multiplayer blocks. Now, the robots can communicate with each other. You can implement the robot team-based applications, not just programming the same robots.

Cliff: [15:07] Boy, that sounds fantastic, great plan.

Akos: [15:10] Yes. That's our next plan.

Cliff: [15:13] Well, I guess you're just leading me into my next question, which is where do you plan to go next with NetsBlox? I know you've received some grant funding at this point, but I also get the sense that this is just the beginning for you. Can you talk about some of your plans for the future apart from robotics?

Akos: [15:28] Yes, that's right. First of all, I'd like to thank the TIPs program (https://www.vanderbilt.edu/strategicplan/trans-institutional-programs/tipshome.php) of Vanderbilt that kickstarted the whole thing. Once we had that going and we had an early prototype, we were able to convince the National Science Foundation to give us some more funding to develop NetsBlox into a full-fledged environment. Yes, robotics is definitely one area we would like to look into.

[15:49] The other one is porting NetsBlox applications to iPhones and Android phones and then give access to the sensors there. Now, the students will be able to write the applications that use the motion sensor [inaudible] or camera even on the phone. That will be another interesting extension.

[16:11] Another, probably longer term, is creating a more advanced version of NetsBlox that, instead of a visual programming language, uses Python. Again, the networking part is still fairly problematic. It's only taught in computer science in college-level classes and not even freshmen college level of classes. Probably junior level.

[16:37] So what if we could have a Python-based environment that exposes high school students to networking, so the more advanced students who find the block-based approach limiting still, they could actually graduate to Python and still use these data sources and the message passing abstractions so that they can write this to the applications.

Cliff: [16:59] That actually leads me to another question. You mentioned that there are limitations to block-based programming languages and that to accomplish certain tasks, you have to step outside of them to text-based languages, say Python, in order to get things done.

[17:12] But could you foresee a future in which, for a certain subclass of students, block-based paradigms meet all their computing needs so that they could stay entirely within a block-based language and still get significant computational tasks accomplished?

Akos: [17:28] I think that again, maybe, it will bring together that whole model-based idea that I was talking about at the beginning and the block-based programming. I can certainly see a domain-specific environment that are not necessarily a general-purpose programming language.

[17:47] These kinds of languages, you can write small programs, few hundred blocks, maybe 1,000 blocks. Beyond that, it becomes unmanageable. Just like today, for example, I teach MATLAB, as I mentioned.

[17:59] That's a full-fetched programming language and it's text-based programming language, yet it's not the general purpose in the sense that nobody's writing operating systems or word processors in MATLAB. That's for very specific scientific computing, numerical computing, sigma processing, and those kinds of applications, and it's great. You can write a few thousand-line codes in MATLAB.

[18:19] It's really a wonderful tool, but it's not a general purpose programming language, so I can imagine these domain-specific environments that will speak to block-based programming because it's simply easier to learn.

[18:34] If your main job is not to write programs, but to do something else, work in the sciences or whatnot, and you find it difficult to learn a general purpose programming language, these domain-specific environments specifically tailored for your domain could be way of the future.

Cliff: [18:50] I guess you might look at a product like Tableau (https://www.tableau.com/), for example, which allows you to do data visualization with a wonderful palette of features, and doesn't require you to step into programming languages, like R or Python, when all you want to do is output an interactive visualization.

[19:05] I could imagine that there'd be tools like that in other domains that will allow you to stay within a block-based language, and to get your work done. There are also tools like Blockly from Google, for example, that allow you to program in a visual style, and then output textual programs in standard languages like JavaScript.

[19:23] Do you see a role for outputting textual programs from NetsBlox in a similar manner?

Akos: [19:28] I see Blockly (https://developers.google.com/blockly/) as somewhat different than the what Snap, Scratch and NetsBlox is trying to do. Blockly is more for people who want to create tools like NetsBlox, so we could have used Blockly instead of Snap, but Snap was much closer to what we wanted.

[19:51] Actually, in one of our projects that we created, a molecular dynamic simulation front-end tool for chemical engineers, and one of the aspects of that tool was based on Blockly, so they could drive this simulation script in a Blockly-based visual programming language. I think for those kinds of things, Blockly is great.

[20:14] For what we're trying to do, I think Snap was much better as a starting point.

Cliff: [20:21] Let's come back around to this theme of computational thinking, because if we're seeing greater adoption of these tools in the K-12 arena, how do you think that'll affect students coming into Vanderbilt?

[20:35] I think one of the things that you and I have talked about in other contexts is the unevenness of people's background in computational thinking, and in experiential programming languages when they arrive on campus and how that maybe shapes their trajectories here. Can you talk a little bit about what you'd like to see?

[20:50] You mentioned you're volunteering in middle schools, I think, or elementary schools. How will this shape the kinds of students that we're going to see, do you think?

Akos: [20:59] That's a great question. There is actually a nationwide movement to introduce more computer science into all kinds of schools, and I think that's great. It's great not because we want everybody to become a computer science major, but basically, most kinds of job these days, be it in sciences, STEM, or even humanities, or anywhere else...

[21:26] A lot of those jobs require some basic knowledge of programming. That's definitely an important consideration. On the other hand, these text-based languages kind of still act as a barrier, in the sense that many kids find it really scary to even take a class that's based on computer science and whatnot.

[21:53] I think one other big advantage of this block-based language is that they lower the barrier and make it more appealing to students, who would otherwise not consider computer science or even just taking a computing course, in general.

Cliff: [22:12] As we wind up, we always ask our guests, because this is an educational podcast, what's their favorite analog educational technology?

Akos: [22:21] [laughs] Analog educational technology, never heard that expression in my life.

[22:28] [laughter]

Akos: [22:30] I was closer in saying drawing in general, but yeah, it's a tool. The calendar, [laughs] [inaudible] , it's probably still...

Cliff: [22:36] If we did a poll, that would be way out on top. Thank you, Akos. This has been terrific, and we really appreciate your time.

[22:49] [background music]

Akos: [22:49] Thank you so much for having me.

Derek: [22:50] That was Akos Ledeczi, Professor of Computer Engineering, here at Vanderbilt University, and Lead Developer of the graphical network programming language, NetsBlox. See the show notes for more information on NetsBlox, including information on how to get started using it.

[23:01] I'm really interested in one of the questions Cliff asked near the end of the interview about the growing emphasis on computer programming in K-12 settings.

[23:08] My younger daughter started learning to code in Scratch last year, in third grade, and she's really loving her after school robotics club this year, in fourth grade.

[23:16] Now, she's choosing for more programming experiences by doing this after school stuff, but I know that Scratch is a standard part of the curriculum at her elementary school, and in others as well.

[23:26] I keep trying to imagine what she'll know about coding, as a college freshman eight years from now, compared to what I knew at the end of my college career, as a computer science major.

[23:36] With more and more programming experiences in K12 settings, it's likely that students will arrive on our campuses in the coming years with some, perhaps, significant skills in computer programming.

[23:46] How might that change the computer science curriculum? How might that change the curriculum in other disciplines? I'd love to hear your thoughts on these questions.

[23:53] You can find this on Twitter @leadinglinespod, and we're going to try this for the first time. You can send us a voicemail via email to leadinglinespod@vanderbilt.edu. I'd love to hear your thoughts on how a K12 programming experiences might change the curriculum at the college level.

[24:12] Leading Lines is produced by the Center for Teaching, the Vanderbilt Institute for Digital Learning, the Office of Scholarly Communications, and the Associate Provost for Digital Learning. This episode was edited by Rhett McDaniel. Look for new episodes the first and third Monday of each month and you can find past episodes on our website, leadinglinespod.com.

[24:28] [background music]

Derek: [24:28] I’m your host, Derek Bruff. Thanks for listening.