What's Wrong With CS Education?

Students begin their computer science education with significant conceptual hurdles to overcome that our education system does not prepare them for. Specifically, these hurdles are in their understanding of how to model real-world problems with precise logical, computational and mathematical terms. Teacher experience has long shown that these concepts can be taught, and that students who learn them have an enormous advantage in computer science education.

Why Don't Most CS Activities Solve this Problem?

The overwhelming majority of coding games or activities children use suffer from what we call 'the trial and error problem.' Digital games, and the digital experience as a whole, encourage young learners to drag, drop and click away until they can find the solution. To gain an understanding of how computers work, students instead need to focus on pre-planning and 'model thinking,' where you think like a computer and step through the solution in your head before executing it on the screen. Most digital coding activities available for children today fail to incentivize that behavior, meaning only the students who already 'get it' will derive real benefits.

How Are Our Games Different?

First, we believe offline, or 'unplugged' coding activities by their nature help to solve the trial and error problem. The effort involved in setting up the board and moving the pieces to physically solve the puzzle teaches kids that unfocused trial and error is the path of most resistance, rather than least resistance.

Our games also fold in core computer science concepts at a far more advanced level than you see in most 'coding for kids' games or activities. Students must work through loops, conditionals, and even Boolean Logic to solve the puzzles in On the Brink, Rover Control and Robot Repair. Robot Repair in particular teaches students core formal logic concepts like 'the inclusive OR' (in formal logic and computer science, 'OR' is equal to our colloquial use of 'and/or'), and how this inclusive OR is negated. In the education system, a student isn't likely to be exposed to this information until college, if at all, and students struggle with these concepts when first exposed to them in computer science class. Unless, of course, they've already learned them, preferably through fun game play.

1321 Cameron Street | Alexandria, VA 22314 | PHONE 703.549.4999 | FAX 703.549.6210

www.thinkfun.com

The "Conceptual Pain Points"

Long experience from computer science teachers has shown several areas where new students, and even experienced ones, consistently struggle – we call these 'conceptual pain points.' The most important of these, which we call the 'master pain point' is the ability to put statements expressed as natural language into mathematical form. (Think of how people tend to struggle to understand double negatives in conversation, for example.)

As another example of learning to think with precision, one critical aspect of writing programs is to properly set your boundary conditions: should you write 'greater than,' or 'greater than or equal to'? What does the program do if you write "NOT (X or Y)"? Developing the ability to understand language as logic, and in mathematical form, is crucial to success at programming.

Our Promise

Our //CODE series games can effectively supplement any programming education because they work at the conceptual level, rather than the level of the programming language. They won't help you learn CSS or JavaScript; instead they'll prepare you to work through the process of coding without falling victim to the problems most students experience. These games help students to exercise and develop their 'logic muscle' through fun game play, explicitly teach the formal definitions of key computer science terms, address key concepts like loops and conditionals, and develop the skill of thinking in terms of 'abstraction and reuse,' the central concept of 'On the Brink.'

Contact Us

Kacey Templin
Public Relations Specialist
703-549-4999 x-3602
ktemplin@thinkfun.com