Unplugging Computer Science to find the science

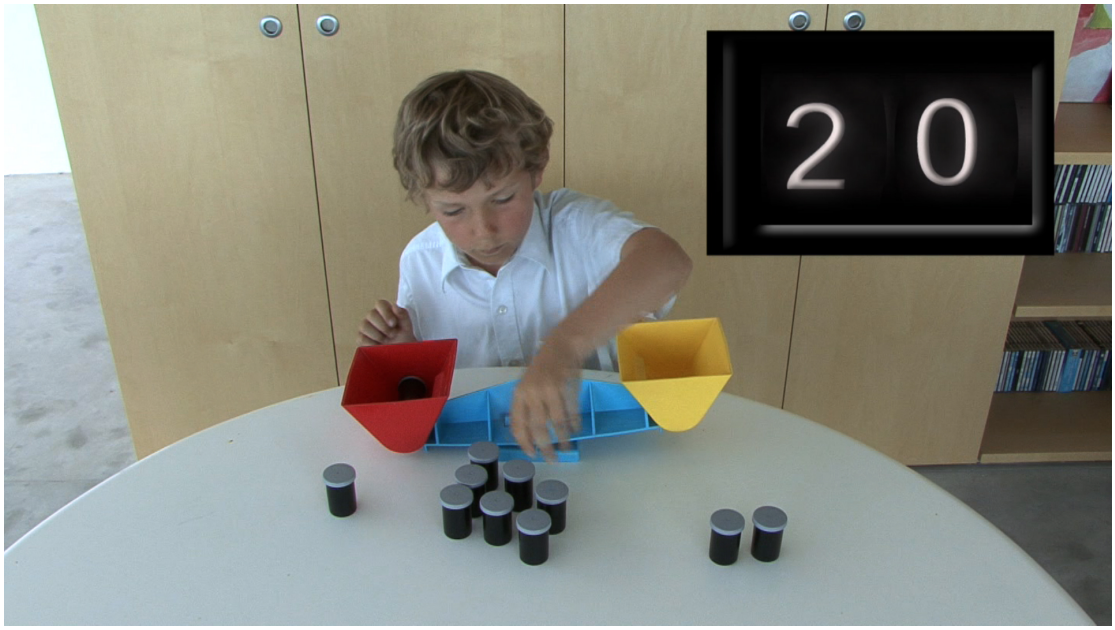
Tim Bell, October 2012

Abstract:

The Computer Science Unplugged project provides activities that enable students to engage with concepts from computer science without having to program. Many of the activities provide the basis of a scientific exploration of computer science, and thus help students to see the relationship of the discipline with science. In this paper we give examples of such activities and how they can be used to engage students with CS as a science.

The Computer Science Unplugged project (csunplugged.org) is a collection of activities that engage young students **(typically 10-18 years old)** with the big ideas of computer science without having to learn to program or even use a computer [1, 2]. This enables us to get to the heart of the subject and dispense with technological distractions. It also exposes the science of computing, since the absence of the computer forces students to learn how the computational principles work by direct observation and experiment. The computer truly is an artifact but not the only means to understand and effect computation. The "unplugged" activities cover a large range of topics – algorithms, human-computer interaction, artificial intelligence, computer graphics, tractability, compression, encryption and more – but *not* learning a programming language or using a computer!

For example, students can explore the concept of the complexity of algorithms by simulating sorting algorithms using a balance scale to sort identical-looking weights from lightest to heaviest, comparing two weights at a time [**see photo 1]**. Students can usually work out for themselves that finding the heaviest weight of, say, 10 items, will take 9 comparisons, and that repeating this process on the remaining 9 will take 8 comparisons. Sometimes we have one student perform the sorting manually, and by the time they have finished their empirical experiment counting the comparisons, the class have analyzed the problem and calculated that it will take 45 comparisons. What we end up with is a very simple example of a process where students have produced their own theory (that the number of comparisons is the sum of the numbers from 1 to n-1), and verified it with an experiment. Furthermore, they are in a position to explore a surprising consequence of the theory: that sorting 100 items this way might be expected to require about 10 times the effort (450 comparisons), but in fact it takes over 100 times as long (4950 comparisons), motivating a quest for better algorithms.



Students can also use an "unplugged" approach to experiment with processes that are less amenable to mathematical analysis. For example, Fitts' law is used in human-computer interaction to determine how long it will take for a user to execute a series of pointer movements (such as clicking on a series of buttons or menus in an interface or touching controls on a smartphone). Fitts' law is typically expressed as

$$T = a + b \log(1+D/W)$$

where T is the time taken to move the pointer, D is the distance moved, W is the width of the target, and a and b are constants that depend on the situation. Students can explore this relationship by simply timing how long it takes to move a pencil back and forth between two targets on paper, and plot this for varying values of D and W [**see photo 2**]. This usually exposes the logarithmic relationship, and also gives them insight into the level of detail that can be put into analyzing what seems like a very simple actions as well as the consequences of design decisions in interfaces like the size and position of buttons. Of course, students in the target age group may not know the concept of a logarithm, but by plotting their results on log-scale graph paper they get remarkable straight lines, which enable them to make predictions by interpolation, and also consider the validity of extrapolation (for example, what if the targets were a mile apart?)

Many of the "unplugged" activities introduce the nature of a problem only (such as the intractability of map coloring or the travelling salesperson problem). Others involve CS-relevant discrete mathematics (such as exploring

the mathematical patterns in binary number representations, or a minimal spanning tree for a graph). Still others explore elements of computing that are impacted by human behavior (evaluating how physical interfaces such as door handles and oven controls might confuse users, exploring the predictability of text, or simulating the Turing Test for intelligence); these activities are based heavily on experiments and heuristics.

Many activities introduce students to the surprises and paradoxes in computing – that it is possible to detect and correct errors in data without having it re-transmitted; that one of the fastest sorting algorithms is slowest when given a list of already-sorted numbers; that a child could design a small combinatorial problem that they know the solution for, but a computer would take billions of years to solve it; and that randomness can help make algorithms run faster.

We have found that when students have used "unplugged" activities for formal learning, they are best supported by a background in scientific method. Usually they will need to describe the purpose of the activity ("experiment"), explain how it was set up (reproducibility), and report on results in a way that shows what they have discovered (well supported conclusions).

Given the wide range of computer science topics that can be explored at length by young students without using a computer, we have a very practical demonstration that the science of computing would exist even if computers did not, and it can be explored using theory and experiment to validate hypotheses. This view is not just a philosophical discussion of semantics; to push the boundaries of what can be done with computing technology, programmers need to understand and stretch the science behind it, and it is such people who can create systems that we might have thought were impossible, or who will prove systems to be impossible before we waste time trying to build them.

Those who can expand the frontiers of the science of computing may not even be particularly interested in programming or the machine itself, but they have much to contribute through their passion and talent for a scientific approach. Such students can be deterred by "computer science" courses that are primarily about programming, and they might even conclude that they don't have ability in the subject. The goal of Computer Science unplugged is to give such students a glimpse of the science of computing as a motivation to learn the methods.

In the Unplugged project we are constantly looking for new activities that will engage young students with a broader range computing principles. New areas that we are exploring include locality of reference (including defragmentation), computer vision, and more. From experience we have learned that it is difficult to predict which games and activities will be engaging, and which are simply become busy work for students, but exploring these ideas with students enables us to refine the ideas, and often it is the students themselves who find ways to frame the activities in a way that is compelling for others to participate in.

[1] Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer Science Unplugged: School Students Doing Real Computing Without Computers. *The NZ Journal of Applied Computing and Information Technology*, *13*(1), 20–29.

[2] Bell, T., Rosamond, F., & Casey, N. (2012). Computer Science Unplugged and related projects in math and computer science popularization. In H. L. Bodlaender, R. Downey, F. V. Fomin, & D. Marx (Eds.), *The Multivariate Complexity Revolution and Beyond: Papers in Honour of Michael Fellows* (Vol. LNCS 7370, pp. 398–456). Heidelberg: Springer.