cs4fn and computational thinking unplugged

Paul Curzon
Queen Mary University of London
Mile End, London E1 4NS
United Kingdom
p.curzon@qmul.ac.uk

ABSTRACT

'Computer Science for Fun' (cs4fn) is a public engagement project aiming to both enthuse school students about interdisciplinary computer science and support computing teachers. It started in 2005, with cs4fn resources now widely used in UK schools as well as internationally. We overview the approach cs4fn has used to inspire students and teachers. In particular we look at how not only subject knowledge but also computational thinking ideas can be taught in an integrated way using cs4fn 'unplugged' activities embedded in stories. We give two illustrative examples, one based on the problem of helping people with locked-in syndrome communicate, the second based around magic tricks.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education—*Computer Science Education*

General Terms

Human Factors

Keywords

cs4fn, computational thinking, unplugged, K-12 education

1. INTRODUCTION

The cs4fn project [4, 6, 2, 3, 14, 13] is a public engagement and outreach campaign based at Queen Mary, University of London run by Paul Curzon, Peter McOwan and Jonathan Black. It began in 2005 as a response to the global collapse in interest in computer science post 2000. It consists of a magazine [9], a website (www.cs4fn.org), live shows [7] and support for school teachers, all focusing on the idea of inspiring students about the subject. The project explicitly goes beyond the school curriculum: we write about and present research topics in accessible ways. This encourages students to learn in 'hobby-mode', helps provide a context for the subject, and shows the subject's exciting leading edge.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
WIPSCE '13, November 11-13, 2013 Aarhus, Denmark.
Copyright 2013 ACM 978-1-4503-2455-7/11/13 ...$15.00.

While the original primary aim of the cs4fn project was to inspire secondary school students, age 14+, about interdisciplinary computer science, that aim has broadened in several ways as the project has progressed. We have extended our resources to younger secondary school and primary school children. Furthermore, originally our support for teachers was indirect, providing magazines and web resources for them to pass on to students, but that they might find interesting and useful to read themselves. We have in recent years introduced a new emphasis on supporting teachers directly, with cs4fn continuous professional development sessions, downloadable activity sheets and other resources directed primarily at teachers rather than their students. This move towards supporting teachers directly has been in part in response to a major change in UK policy away from a curriculum focusing on the use of technology to one on computer science being taught at all stages from primary to secondary school. There is however a skills shortage, with teachers being required to rapidly adapt to the new curriculum. Similar issues face other countries within their own local contexts.

2. THE SUCCESS OF CS4FN

The cs4fn project has been highly successful on a range of measures. For example, demand for physical copies of the magazine is strong and has increased steadily. We send individual copies to 5400 schools across the UK. Over 1000 schools or individual teachers have then in addition subscribed to extra copies (e.g., class sets). These are used in a variety of ways: with 'gifted and talented' groups and with 'problem' classes, to use directly in class and to be left for students to read in break times, and even as a different kind of reading material placed in literacy boxes. We have also had requests for physical copies from subscribers in over 80 countries. In a survey of teachers conducted in 2012, 98% rated the cs4fn magazine as either "excellent" or "good". 77% agreed that they could use articles or ideas from cs4fn in their lessons.

The website has been similarly successful. Between 2008 and 2013 it has received over a million visits and PDFs of our magazines and other resources have been downloaded over 890,000 times. Web users have been positive in surveys with over two thirds of respondents saying cs4fn helped them see more ways computer science is used in the real world. After visiting the cs4fn website, they reported thinking of computer science as more interesting and thinking of a variety of careers that would use computer science.

Our series of shows in schools have also been very popular with talks given to nearly 20,000 school students in some 270 visits to schools and universities and over 10,000 more through science festival stalls. Teachers have been highly positive of these school talks in follow-up surveys. With 100% of those surveyed, for example, saying our live shows met their needs and that they would recommend them to others. These surveys also suggest students are more likely to take computing courses as a result.

The success of cs4fn is due to a variety of reasons, though the idea of telling research stories in a fun way is key. Having a clear and fun narrative, with interesting twists and links, is central to all we do. We use metaphors a great deal as a form of scaffolding to build on everyday non-computing ideas of which people are already aware from everyday life. In particular, we extensively use non-computer based examples of computation to illustrate concepts. This extends naturally to our talks in the form of kinaesthetic 'unplugged' activities where computation is acted out in various forms away from computers, making abstract concepts both tangible and visible [10]. In general we do not do unplugged activities in isolation but embed them within a narrative in talks. This approach gives a way, not only to deliver subject knowledge, but also to introduce the concept of computational thinking [16].

3. COMPUTATIONAL THINKING

Jeanette Wing [16] argued that computational thinking is a set of transferable skills developed while studying computer science that are different to that developed more generally or around other disciplines. The idea has become extremely popular, and has now been included as part of the UK school curriculum. Core concepts within the computational thinking skill set include algorithmic problem solving, logical thinking and the ability to use a variety of forms of abstraction.

If computational thinking is an important skill set to be gained by studying computing, as Wing has argued, the question arises: how do you best teach it to school children? [17] One answer is that these are skills that you just pick up by studying the subject. As you learn to program, write more programs, do a course in data structures and algorithms, and so on, you will gradually pick up algorithmic problem solving skills. Logic and proof courses in addition will improve your logical thinking and rigorous argument skills, and so on. This is true up to a point, but if the skill set is so important then as teachers we should be actively helping students develop that integrated skill set. One of the issues with this as an answer is also that it suggests that computational thinking as a whole is developed fairly late on. It is very hard to gain a deep understanding of the issues from writing small programs or doing simple logic problems, for example.

We are interested in how one might introduce the overall idea of computational thinking, both in schools and to teachers themselves, and especially at an early stage when students may have little actual computing experience. Rather than introducing the separate skills in some order we believe it is important to see how the whole coherently fits together. If people understand the combined skill set from the start, they will be able to develop those skills more effectively.

We are therefore particularly interested in exploring ways to introduce computational thinking itself as a coherent skill set, before the skills embodied in the elements have been fully developed. Is it possible to introduce the ideas to complete novices who cannot yet even program? Can it be introduced to younger children, including those who will not do computer science in any depth in their later studies? A further issue is the importance of introducing the idea that computational thinking is not actually just about computer-based problem solving. If the skills are only developed in a computer science context then this may easily be lost.

Based on these ideas, we have been introducing computational thinking using the cs4fn approach of making things fun using research-driven storytelling together with unplugged activities, as we have previously successfully done with subject knowledge. We hope in particular that we can give teachers a deeper understanding of computational thinking, as well as giving them concrete activities that they can use in their own lessons.

We have focused on a series of sub-skills from within the computational thinking family, but aimed to show how they combine and interact together. In particular we have concentrated on the following: algorithmic problem solving, logical thinking, analysis of the efficiency of solutions, rigorous argument and proof, translating solutions between domains, computational modeling, abstraction and understanding people.

The latter point concerning understanding people is not always focused on in the computational thinking set of skills, however we believe it is an incredibly important transferable skill that computer science students should and do develop as part of human-computer interaction strands of their studies, for example. Computer scientists solve problems, but those solutions are ultimately for people, so must be designed to work for people.

4. EXAMPLES OF THE CS4FN APPROACH

Unplugged activities are typically used to both inspire students about computer science and teach subject knowledge: whether binary numbers, human-computer interaction principles or error-checking codes. They can also be used to introduce computational thinking concepts to novices giving an understanding of how the different skill elements work together. The links from the activities to computational thinking concepts need to be explicitly drawn out. Embedding the activities in stories helps to give the bigger picture. In the following subsections we overview two (of many) examples we have used to introduce these ideas explicitly to teachers as part of continuous professional development sessions. Our aim is to both illustrate the cs4fn approach generally and show how direct links to computational thinking ideas can be made using it.

4.1 Locked-in Syndrome

Our first example is based on the problem of helping someone with locked-in syndrome. In its full version with students we do this as an hour long interactive lecture session. We give a compressed version for teachers. A full version of the 'story' we tell and how it links to computational thinking is given in [5]. Here we give an overview.

Locked in syndrome is a medical condition where a person is totally paralysed. If lucky they can blink one eyelid. In his autobiography [1] Jean-Dominique Bauby describes life with locked-in syndrome. He wrote the book only able to blink one eye and without any technological help. He just had a human to write down his words. We explore with the audience how he did this. The helper read through the letters of the alphabet until Bauby blinked then wrote that letter down. As a simple unplugged activity we get the audience to communicate a message to their neighbour in this way, and then explore with them issues that have to be solved to make it work. The audience discover, for example, the need for a backup mechanism to correct errors, and a way to deal with numbers and punctuation. They also normally suggest ways to improve it such as predicting words from the first few letters. At this point we can discuss algorithmic problem solving in a situation where getting the detail right matters, even though we are not talking about computers. We also immediately have an example of translating solutions in that predicting words is all predictive texting does - a solution for dealing with the difficulty of typing on a small device is applicable in this context too.

We then explore how much effort must have been needed to write the book. We introduce an abstraction at this point. Rather than worrying about actual times we use a measure of work instead: the number of letters of the alphabet spoken (i.e., questions asked) by the helper. We then talk through a simple best / worst / average case analysis based on communicating an individual letter. We have the basis to discuss ideas of both abstraction and analysis of efficiency even with novices.

We next point out that rather than the 13 letter average case of this method, it is possible to do it in 5 questions in the worst case. A show of hands implies most don't know how this could be done. We suggest everyone knows the kind of question to ask if we switch problems. We then do another unplugged activity - playing the children's game of 20-Questions where they have to work out which famous person someone is thinking of by only asking yes-no questions. They are encouraged to think about the kind of question people are asking and what makes good questions. From the start they ask questions that eliminate half the possible answers, and can explain why those are good questions. This leads to a discussion of search algorithms, divide and conquer, and more analytical thinking about its efficiency - from a million famous people we could know who it was in 20 perfect questions as shown by repeatedly halving a million. We also have another example of translating problems as now we can apply this solution to our original problem of working out a letter someone is thinking of. Only at this point do we bring in ideas for technological help - how might you replace the human helper with a computer system that implements the algorithm we have come up with?

The audience at this point generally agree that with some computer science / computational thinking we have made Bauby's life better. We end with a final twist - were we counting the right thing? Bauby would now have to blink up to 5 times instead of just once per letter. We might have made things better but if blinking is hard for him we may have made things worse. We should have found out first. Ultimately our problem solving is for people so we have to be sure our analysis and solutions fit their actual problems.

Compressed into this one problem solving story we can illustrate a variety of different aspects of computational thinking, and also how they work together. This is a very accessible way to introduce the concepts and how they can be used to solve a real world problem. All this has been embedded in a story with no technology and no code involved until the end in a way that novices can follow.

4.2 Magic

A second way to introduce computational thinking is a computer science magic show [7, 15, 8]. In addition to giving live shows we also have 2 free to download magic of computer science books [11, 12] and a major area of the cs4fn website (www.cs4fn.org/magic/) devoted to magic where each of the tricks outlined below are described in full. The idea is to use magic tricks to illustrate computing concepts. We do a real magic trick and then challenge the audience to work out how it works, before explaining the secret. Finally we explain linked computing concepts. We have developed a wide range of trick based lessons around this format, illustrating a variety of concepts including what algorithms are, search algorithms, error correcting codes (adapted from a CS Unplugged activity), why rigorous testing is important, formal verification, usability and user experience.

The key is that any magic trick consists of a secret method (an algorithm) and a presentation (the interaction design). Unless both work a magic trick will not work. The link between magic method and computer algorithm is not just a metaphor. Some magic tricks even use identical algorithms to computer algorithms. For example, one trick allows the magician to reveal a card hidden in an envelope throughout the trick that is the same as one selected by a member of the audience. The secret mechanism is just a simple search algorithm. The mechanism of another trick that allows a blindfolded magician to know exactly what card in a grid was turned over is a parity-based error correcting code. This gives an aspect of computational thinking that can be introduced immediately: that of translating solutions between domains. One algorithm can work in domains as apparently different as predicting a chosen card in a magic trick and computers searching for data.

Inventing new magic trick mechanisms involves the same skill as inventing new algorithms. This gives a way of showing what algorithmic thinking is, separate from coding itself. With a magic trick you also have to specify the steps precisely and in the right order even though they are for a human not a computer to follow. They must work whatever happens: whatever card is chosen, for example.

The fact that presentation matters in magic also gives a natural way to introduce the importance of understanding people in creating algorithms. In fact the same understanding of cognitive psychology is needed to make a trick work well as make a program usable. Tricks give a powerful way to demonstrate how the way a system is engineered can affect whether people make mistakes or not. Magicians control people's attention to make them all make a mistake at the same time, a computer interface needs to be designed to control their attention so they do not miss things. Understanding people matters.

No one wants to stand in front of a live audience and do a trick that might not work. Tricks gives a natural way of introducing the idea of testing and also how it can be insufficient. This leads to logical thinking and how it can lead to a reduction in test cases needed to be sure a trick will work, and then a discussion of how rigorous argument and proof can be used to prove the correctness of a trick (or algorithm). Ideas of mathematical and computational modelling can also be introduced. There are various ways that tricks can be argued to be correct. Some involve mathematical notation for appropriate audiences but maths can be avoided in the discussion (e.g., using diagramatic proof).

The arguments used also give a way to introduce abstraction. For example, in a proof of one trick we abstract away from their actual values and refer to the cards involved by their original position as a way to simplify the argument.

Magic tricks thus can give a powerful way to introduce a range of computational thinking concepts. Creating a trick that you are sure works requires bringing all these concepts into play. This can be done with an audience with no computing experience at all. It can be done around a single trick or a series of different ones each emphasising a different aspect. We give a whole magic show embedded in a story around our own research on the design of safer medical devices. The magic tricks are used to explore the issues in ensuring such safety critical computer devices work correctly and those using them do not make mistakes, for example in delivering correct doses of drugs. The magic show also illustrates that not only computer scientists but also magicians need to develop computational thinking skills.

5. CONCLUSIONS

cs4fn has been a highly popular way of enthusing both students and teachers about computing, introducing subject knowledge in a fun way that both reinforces and goes beyond school syllabuses. More recently we have used the same approach of research based fun storytelling and unplugged activities to explicitly introduce the idea of computational thinking. We have outlined two examples (of many) we have used to illustrate the approach.

Feedback about the cs4fn style of session has been extremely positive from teachers both about sessions for students and for teachers themselves. For example, a survey-based evaluation of one recent 2-hour continuous professional development workshop adopting the approach outlined above had 100% positive satisfaction ratings from the teachers (20 responding from 48 attending) with all agreeing it had given them useful ideas for the classroom.

6. ACKNOWLEDGMENTS

cs4fn was funded by EPSRC (EP/F032641/1) (2008-2013) with additional support from Google's CS4HS programme. This paper was written with support from the CHI+MED project, funded by EPSRC (EP/G059063/1). cs4fn was created by Paul Curzon and Peter McOwan. Peter McOwan is the inspiration behind the magic of computing. Since 2008, Jonathan Black has helped ensure its success along with Jo Brodie and Chrystie Myketiak. Laura Meagher has helped enormously as our external evaluator. Nick Cook, of the School of Computing Science at Newcastle University, organised the teacher CPD session and evaluation mentioned.

7. REFERENCES

[1] J.-D. Bauby. *The Diving-Bell and the Butterfly.* Fourth Estate, 1997.

[2] T. Bell, P. Curzon, Q. Cutts, V. Dagiene, and B. Haberman. Introducing students to computer science with programmes that don't emphasise programming. In *Proceedings of ITiCSE 2011, The 16th Annual Conference on Innovation and Technology in Computer Science Education ACM SIGCSE*, page 391, June 2011. Darmstadt, Germany.

[3] J. Black, P. Curzon, C. Myketiak, and P. W. McOwan. A study in engaging female students in computer science using role models. In *Proceedings of ITiCSE 2011, The 16th Annual Conference on Innovation and Technology in Computer Science Education ACM SIGCSE*, pages 63–67, June 2011. Darmstadt, Germany.

[4] P. Curzon. Serious fun in computer science. In *Proceedings of the 12th Annual Conference on Innovation and Technology in Computer Science Education ACM SIGCSE (Invited Keynote)*, page 1. ACM, September 2007. Dundee, Scotland.

[5] P. Curzon. *Computational Thinking: Searching to Speak.* Queen Mary, University of London, 2013.

[6] P. Curzon, J. Black, L. R. Meagher, and P. W. McOwan. cs4fn.org: Enthusing students about computer science. In T. O. C. Hermann, T. Lauer and M. Welte, editors, *Proceedings of Informatics Education Europe IV*, pages 73–80, November 2009. Freiburg, Germany.

[7] P. Curzon and P. W. McOwan. Engaging with computer science through magic shows. In *Proceedings of ITiCSE 2008, The 13th Annual Conference on Innovation and Technology in Computer Science Education ACM SIGCSE*, pages 179–183. ACM, June 2008. Madrid, Spain.

[8] P. Curzon and P. W. McOwan. Teaching formal methods using magic tricks. In *Fun with Formal Methods: Workshop at the 25th Int. Conference on Computer Aided Verification*, July 2013.

[9] P. Curzon, P. W. McOwan, and J. Black. *cs4fn: Computer Science for Fun Magazine.* Queen Mary, University of London, 2005-2013.

[10] P. Curzon, P. W. McOwan, Q. Cutts, and T. Bell. Enthusing and inspiring with reusable kinaesthetic activities. In *Proceedings of ITiCSE 2009, The 14th Annual Conference on Innovation and Technology in Computer Science Education ACM SIGCSE*, pages 94–98. ACM, July 2009. Paris, France.

[11] P. W. McOwan and P. Curzon. *The Magic of Computer Science.* Queen Mary, University of London, 2008.

[12] P. W. McOwan, P. Curzon, and J. Black. *The Magic of Computer Science II: Now we have your attention.* Queen Mary, University of London, 2009.

[13] L. R. Meagher, P. Curzon, P. W. McOwan, J. Black, and J. Brodie. *cs4fn Final Evaluation Report.* Queen Mary, University of London, In press.

[14] C. Myketiak, P. Curzon, J. Black, P. W. McOwan, and L. R. Meagher. cs4fn: A flexible model for computer science outreach. In *Proceedings of the 17th ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, pages 297–302, 2012.

[15] C. Myketiak, P. Curzon, P. W. McOwan, and J. Black. Teaching HCI through magic. In *The Contextualised Curriculum: A CHI 2012 workshop*, May 2012.

[16] J. M. Wing. Computational thinking. *Communications of the ACM*, 49:33–35, 2006.

[17] J. M. Wing. Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical Physical and Engineering Sciences*, 366(1881):3717–3725, July 2008.