



SWITCHED ON

COMPUTING AT SCHOOL NEWSLETTER

AUTUMN 2016

COMPUTING FOR ALL

INSIDE THIS ISSUE

CAS COMMUNITY p2-5
CAS NI goes from strength to strength, awards for CAS Scotland, our 200th hub and a major new CPD project.



COMPUTING FOR ALL p6-14
A major feature with many contributions from colleagues at the forefront of challenging inequality and developing an inclusive curriculum. Practical ideas and articles featuring the pioneering work of CAS #include.



TEACHING AND LEARNING p15-19
Mark Thornber continues his Mathematical Musings, Dave White presents a new course exploring the pedagogy of CT and a new free book from Peter Millican using illustrative computer models in Turtle System.



COMPUTING THEORY p20-23
Greg Michaelson considers some benefits of the GOTO command, John Stout looks at what computers can and can't do and Paul Revell sings the praises of a book explaining the inner workings of a processor.



Ideas, particularly good ideas, can take a long time to gain traction. Take the notion of Computational Thinking (CT), a term first coined by the late Seymour Papert. Papert was pointing to the potential of new technology to facilitate children's ability to solve problems and thus 'construct' knowledge and understanding. But it took many years for the term to enter more mainstream use. For that we can thank Jeannette Wing. In a short paper (goo.gl/uRP3AI) written in 2006, the professor, then at Carnegie Mellon University argued that "Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every

child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking." She pointed out that "Thinking like a computer scientist means more than being able to program a computer", going on to stress that "This kind of thinking will be part of the skill set of not only other scientists but of everyone else. Ubiquitous computing is to today as computational thinking is to tomorrow. Ubiquitous computing was yesterday's dream that became today's reality; computational thinking is tomorrow's reality."



Jeannette Wing

Too many people still see CT as something for the technically minded. CAS takes a different view. CT has a generic value for developing ways of thinking in *all* children. The benefits are applicable to many areas, not just Computing — one reason CAS lobbied for a curriculum entitlement across all key stages. This issue focuses on inclusion; on making Computing accessible to *every* child, not just a select few.



DEVELOPING A NEW GENERATION OF CURRICULUM LEADERS

John Woollard, co-ordinator of the CAS Tenderfoot project, and teaching fellow at the University of Southampton outlines the objectives of an exciting new initiative aimed at secondary school teachers.



TEN UNITS TO EXEMPLIFY COMPUTER SCIENCE

Each Unit typically involves around 10 activities which can form the basis of separate shorter CPD sessions. These will be introduced over the coming year. Others will follow.

- **A Conceptual Approach To Programming** is probably the most familiar territory. Split into two separate sessions, the first has an emphasis on laying firm foundations. The second considers introducing data structures.
- **Clever Stuff For Common Problems - Going beyond simple algorithms** builds on an appreciation of data structures to introduce some real world algorithms.
- **Bits and Bytes - The digital advantage** looks at the representation of different data.
- **Theoretical Computer - Fun with finite state machines** considers models of computation and Turing Machines.
- **Bits AND Chips - The simple ideas that make computers tick** explores Boolean logic and machine architecture.
- **Simulating Our World - Adventures in agent based modelling** gives practical ideas about 'computational abstractions'.

Others planned are **Doing Stuff and Doing It Well** (the search for clever algorithms), **Thinking Machines** (the quest for artificial intelligence), **Communication Basics** (the clever ideas that made the internet) and **Clicks and Mortar** (making sense of the way the web works). Watch the video about the Tenderfoot project at goo.gl/glyhQk.

CAS Regional Centres are starting to draw together the activities of CAS curriculum champions such as Hub Leaders and Master Teachers. At the heart of this work is our belief that face to face discussion, sharing ideas and support are key to developing local communities of practice. Over the coming years we hope many secondary teachers will benefit from the Google funded CAS Tenderfoot project. It provides high quality, subject deep and resource rich CPD for these curriculum champions exemplifying Computer Science at Key Stage 3. Each fast-paced, full day unit aims to introduce potential trainers to a body of theory (of broadly A-level standard) and a range of ideas for introducing these to less experienced colleagues in their local CAS communities.

As such, the project is focused on utilising the local CAS community infrastructure (Master Teachers, University academics and Hub Leaders in particular) to develop secondary school teachers who have little or no background in Computer Science. This is a long term project, the first stage of which is to develop a network of Tenderfoot Trainers who can offer the one-day sessions to experienced

teachers in their locality. The

resources supporting each day are structured to allow those teachers to then offer shorter sessions, ranging from brief inputs at departmental or CAS Hub meetings to half-day or twilight training sessions to help empower teachers to continue to develop their curriculum from Key Stage 3 up. Each one day session gives access to a range of materials including a comprehensive presentation, detailed trainer's notes, all related resources for the classroom activities and supporting teachers' notes. That said, the focus is on developing teachers, not simply providing classroom resources. Much of the subject matter will be unfamiliar and we hope to promote discussion and debate about how best to introduce Computer Science concepts to pupils as well as provide teacher friendly exemplars.

The materials draw on many existing resources, including those shared by teachers on the CAS community resources. In each session these are developed around a narrative that seeks to illustrate the centrality of core concepts in computational thinking such as algorithms, abstraction, decomposition, generalisation and evaluation. All materials are available under a Creative Commons license to encourage further development and sharing in line with the CAS ethos. The project also promotes an understanding of the professional values of research and its implications for continuing professional practice.

You can find more details of CAS Tenderfoot CPD events or becoming a Master Teacher through your Regional Centre: goo.gl/rthUJL. If you'd like further details about becoming a CAS Tenderfoot trainer please contact Tenderfoot@computingatschool.org.uk

The collage contains several educational elements:

- Finding The Shortest Path:** A graph with nodes and edges, highlighting a path.
- Discovering Termite Rules:** A diagram with a 'Breadth First' label and a box asking 'To keep complex tasks simple?'. It includes a 'Random movement' box and a 'Decomposition' callout ('breaking down into parts').
- The structure of language:** A parse tree for the sentence 'The cat sees Janet'. The root node S branches into NP (the) and VP (cat sees Janet). VP branches into V (sees) and NP (Janet). The NP 'Janet' further branches into PN (Janet).
- Other callouts:** 'Storage: Create variables and data structures', 'Inputs: Size of data structures', 'Proc: Compare things', 'Increment: increment (or reset to zero)', 'Appropriate: appropriate', 'Evaluation: making judgements', and 'Generalisation: spotting & using similarities'.



EVERYONE BENEFITS FROM JOINT

WORK AND CO-OPERATION



CAS Hubs are the lifeblood of our community. Last term Laura Pearce, then Head of Computing at St-Luke's Science and Sports College in Exeter, reflected on what it meant to host a CAS Hub.

We are fairly new to being a Hub School and we wanted to ensure we started well. At St Luke's we tried to cater for both Primary and Secondary colleagues from across the city. We had excellent links with our feeder primaries through a community PE project, so making contact was easy. We're also in a convenient place with great transport links, so reaching our school was straight-forward.

We tried really hard over the past two years to ensure our scheme of work reflected changes to curriculum and also Ofsted requirements. We had to ensure we had a well-rounded curriculum that incorporated some primary work for students who hadn't been taught Computing before. I felt it was important to share with our Primary colleagues what we do in Computing when their students reach us. Having a good transition between the two is so important.

Our Hub meetings have been well-attended. We had colleagues from St Luke's attend out of interest, plus both local Primary and Secondary teachers. Our local FE College also expressed an interest, which is great as we only cater for 11-16, meaning that the majority of our students go on to FE College in the city. I also tried really hard to rope in (encourage!) colleagues from further afield. Devon is so spread out that often a school can be relatively isolated, particularly more rural ones. This created excellent working partners and many friends.

We are close to The Met Office and they were brilliant, helping us with both resources and speakers. We now have a lovely relationship with them. They have spoken and attended all our Hub meetings and they been in-

credibly generous with their help. We have also had SWGfL (South West Grid for Learning) in to speak, as well as teachers from other schools. There is no competition which is so refreshing; everyone simply wants to share good practice.

We have always provided good refreshments too! At the end of the school day, I am very aware that people are using their own time to come and see us – making them feel welcome is very important. We did a raffle last time. It encouraged people to stay to the end and they were appreciative. Local businesses have donated and a bottle of wine seems very well received! At the last meeting, I allowed time for 'networking'. I hate this phrase and was mocked greatly by my colleagues, but I feel it was one of the most useful parts of the meeting. It meant people walked around our lovely Theatre, talking to one another. Many 'followed' each other on Twitter and were delighted to meet in person! I know it created many useful relationships that are still ongoing and I love the fact that we helped create that.

I love being a Hub Leader, knowing we have broadened our knowledge, but also helped others too. We all have the same goal and it's been brilliant to work alongside others who share the same aim.

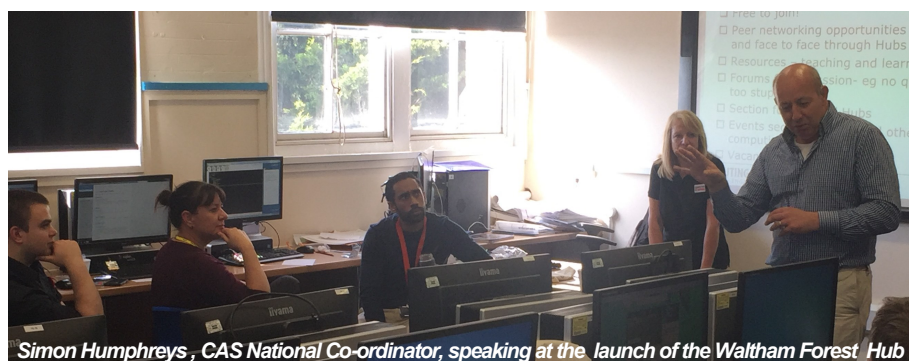
CAS HUBS REACH TWO HUNDRED MILESTONE

In the last few years CAS has grown into a vibrant community committed to developing Computing in schools. It is recognised around the globe as a model for curriculum development. The CAS Hubs are the foundation of much of this work.

Education is an ever changing landscape these days. However, one constant in that landscape are the teachers: dedicated professionals committed to inspiring their pupils and developing their own understanding. Over the past three years the number of Hubs has grown significantly and our 200th Hub was launched last July at Norlington Boy's School in Waltham Forest, London.

The new Hub Leader, Demetrios Skamiotis commented, "We are very excited about the launch. It will provide vital face to face support for teachers – giving them a relaxed, informal place to meet and share ideas, resources, receive training and get up to date information, advice and support. Our meetings will be run by teachers - for teachers. We are planning a lively programme of activities and look forward to welcoming local teachers to the group."

If there isn't a Hub near you, why not follow the lead shown by Demitrios and Laura (left)?



Simon Humphreys, CAS National Co-ordinator, speaking at the launch of the Waltham Forest Hub

SCHOOL BASED RESEARCH IN COMPUTING EDUCATION

For the second year running we held a research stream at the CAS Teacher Conference in Birmingham. Three sessions were led by teachers on the Teaching Inquiry in Computing Education (TICE) project. A booklet of their work is available at goo.gl/ODzHYM - please do take a look if you haven't seen it. The CAS TICE project was a pilot investigating how to support teachers setting up action research projects in Computing. The underlying motivation was a belief that teachers carrying out their investigations in school would value

support from academics in the design and analysis of their study. The investigations support their professional development, have an impact on teaching and highlight further research areas. Funding for the project was provided by Google.

Over 20 teachers and 7 academics have been voluntarily involved since October 2015. The focus of the first meeting was to introduce the basics of designing a research intervention, and to establish and focus on specific research questions. A second meeting, last March was designed to find out how to analyse data gathered and how to write this up.

We were keen to provide teachers with easy ways of sharing and disseminating their research projects, however small, and have created a template "poster", used to create posters, presentation and a team booklet summarising the research. This will be fed back to the teachers' schools, who have released them to participate in the project. Those of us who helped on the project were incredibly impressed by the enthusiasm and energy of the teachers engaging in this project, despite their lack of time to work on their research during the normal school week. *Sue Sentance*

CAS SCOTLAND PROFESSIONAL LEARNING PROJECT AWARD

An innovative CPD curriculum project recently won a prestigious award from ScotlandIS, the trade body for the digital technologies industry. Chair of CAS Scotland Kate Farrell reports.



Professional Learning and Networking in Computing (PLAN C) is a Scottish Government funded project that has developed and delivered high quality professional learning to Scottish Computing Science schoolteachers. The project won the Best Education Provider / Training Programme, Digital Technology Award. The aim of PLAN C was to support all CS teachers to deliver new enhanced qualifications as part of Curriculum for Excellence. "PLAN C has revolutionised the provision of continuous professional development for Computing teachers" said Professor Alan Bundy, from the University of Edinburgh. "It has used the latest, evidence-based pedagogy to empower teachers to provide the kind of Computing education that the world needs in the 21st century. It combines the teaching of programming with the ability to think computationally to increase students' problem solving abilities."



PLAN C focused primarily on the development of computational thinking skills required for creating and understanding solutions in programming, web, and database languages, since it is these skills that are typically so hard to foster. Traditional computing teaching often introduces problem solving too early, leading to cognitive overload, which leaves no room for any real learning. Computational systems, such as programming languages, are normally introduced by example, with novices trying to write programs before they've learned how to read the language. One 'lead' teacher commented, "It is simply not good enough to assume that programming is hard, programming cannot be taught and some pupils simply cannot program. PLAN C confirmed what I had always believed: if the content is presented in ways that are accessible, then all pupils can achieve."

PLAN C instead focuses on helping novices to learn how to talk about key concepts and develop clear mental models. We developed a series of approaches to teaching, using research findings in a way that is relatively easy to adopt in the classroom. We trained a network of 50 'lead' teachers from secondary schools all over Scotland. We supported them to create 25 local teacher hubs spanning the country. At least 350 of the 650 secondary CS teachers in Scotland have been involved in the programme. PLAN C independent evaluator Laurie O'Donnell said, "The quality of the professional conversations I have witnessed has been of an exceptional standard as teachers grapple with the challenge of applying research on CS specific pedagogy in their classrooms."



NORTHERN IRELAND CONFERENCE GETTING BIGGER EVERY YEAR

The 3rd CAS Teacher Conference for Northern Ireland was a sell out with over 130 people attending. Co-organiser and leader of CAS NI Irene Bell reflects on another stimulating day and the ever growing CAS community in Northern Ireland.

The number of delegates attending the conference has nearly tripled in the two years since the inaugural CAS Conference, showing the momentum that has gathered. A wonderful, positive atmosphere prevailed throughout the entire day. LEGO Education brought the child out in all of us and set the scene at the keynote session with a little bit of computational thinking through the 'duck challenge'.

Delegates were treated to choosing from fourteen different workshops with something for everyone. We had drones being controlled through iPads in one room while Stephen Howell introduced us to the 'Internet of everything' in another. Code club sessions were offered for those who were starting the computing journey while more experienced educators enjoyed workshops provided by C Shark targeting A-level teachers.

This year schools presented poster sessions at registration and morning coffee demonstrating and encouraging

colleagues on what can be done. It was great to see this new initiative. One teacher commented "I bumped into a friend with whom I had graduated 31 years earlier but had lost contact and there we were at a computing conference sharing ideas for our classes". Another teacher followed up on the conference the following week with an email indicating that she had taken 3 ideas from the conference and would be using them during the next academic year.

The CAS Conference continues to offer a rare networking opportunity for educators and stakeholders in Computing education in Northern Ireland. The workshops annually present refreshing ideas for teachers to incorporate into their own teaching. The CAS name and the annual Conference are now firmly established within Computing education in Northern Ireland. We look forward with optimism to future CAS conferences in Northern Ireland, which we hope will continue to grow and expand.

POST 16 STUDY HELPS FOSTER INDUSTRY LINKS

St. Cecilia's College Derry have introduced the new Software Systems Development course in Year 13 to a select group of pupils who have shown interest in programming at A-level with the possibility of studying Software Engineering or similar courses at university. Prior to this introduction there were few choices in Computing courses offered to post-16 pupils. It has also helped facilitate the introduction of the new CCEA Digital Technology course at GCSE.

This course was designed by IT industry giants in collaboration with CCEA. Teachers are provided with excellent opportunities to learn C# with weekly tutorials delivered by lecturers at University of Ulster, Magee prior to teaching and an intense week provided by AllState NI and their trainers at the end of June. Along with this a weekly mentoring program was established between AllState NI and the teachers who took on the teaching role, thus building and cementing good relationships between the industry and schools.

WELLINGTON COLLEGE WIN ALLSTATE YOUNG SOFTWARE ENGINEERS AWARD

Last term 10 schools from the Greater Belfast Area were selected to compete in a web-based challenge, sponsored by W5 and Allstate, in Strathern School. The theme, "Creating a Blue Society" focused on the sustainability of the ocean. Prior to the event each team was to build their own website and were encouraged to introduce more advanced elements. Teams then attempted to recreate the website they had been developing to demonstrate their newly acquired programming skills on the day.

Prizes were awarded based on a range of criteria. The Wellington College team of Harvey Duffin, Aaron Burke, Joshua Barret and Paddy Boyle and Mr Lyttle won the "Allstate Young Software Engineers Team Award for 2016 for Key Stage 3". Harvey Duffin was also singled out for a special award for his excellent video/image production for the website.



The judges commented that the Wellington College website was outstanding and more of an A level standard than KS3. They were most impressed with the overall professional design, the high quality photos/video assets and the mobile friendly features of the website. A fortnight later the same team won the Beltec Gaming Competition sponsored by Kainos with an action game coded in Python.

STARTING TO MAKE AN IMPACT LORNA ELKES' DIARY



What a difference two years makes! At times the move forward has felt frustratingly slow but as I look around our classrooms now, I can't believe how far we have come. Last term we reached a land-

mark - the ancient resource server held together by string and hope was turned off! Our resources are cloud based and readily available to review, edit and share regardless of device, platform or browser.

The yellow, tattered and often AWOL reading records have become redundant for most of our students. Initial teething problems have been overcome quickly and pupils are far more engaged in their reading. Older pupils are also recording their ideas and thoughts about a book, not just the page number they got to.

We continue to build pupils programming skills. Phil Bagge's fantastic resources (code-it.co.uk) have enabled the older pupils to accelerate their understanding of Scratch, using it to program a robotic arm. Even better - I'm not leading this. Having greater confidence in the technology has enabled staff with a healthy interest in it to become far more proactive.

I am in awe of some staff and in the fantastic position of going to them for advice. Recently Year 5 pupils enjoyed video conferencing (via Google hangout) with our Chair of Governors who was unable to come into school to be interviewed. The headteacher and I watched in delight whilst the children took in all in their stride as if it were a daily occurrence - no doubt for them it will be. The Chromebooks are an overwhelming success - next is to invest time into developing our other devices to the same high standard. Year 1 have made significant use of the small number of iPads we have. To broaden our children's digital experience we are also beginning to look at Android tablets as well as non-Google laptops. This was always part of the plan, to ensure pupils did not become device specific learners, but had a deeper understanding of the possibilities offered through technology.

FIRM FOUNDATIONS TO TACKLE GENDER IMBALANCE



Phil Bagge, a Hampshire Advisory Teacher spotlights the potential impact the 'hidden curriculum' can have in shaping the outlook of primary school pupils.

Approximately 96% of the algorithms and programming that govern our digital devices are written by men. I wonder how our modern digital world would look and feel if women had a more equal say in designing and creating it? So how can we create a solid foundation in primary education that leaves all pupils excited about computing? We need to stop solving things for pupils and we need to stop them solving things for each other. If one pupil solves something and shares the solution with their neighbours then only the original problem solver develops their thinking skills. The helper is not really helping, in fact they are making the helped more helpless and less independent. Hints rather than solutions help someone to help themselves. Removing the false help of a fully formed solution forces pupils to think for themselves and develop their own thinking skills. Our quieter less confident pupils of both sexes flourish when the tyranny of false help is removed.

I lead lots of computing inset in schools across England and in nearly every school, before I start, teachers come up and tell me, often with a certain level of challenge, that they are rubbish at using technology. When team teaching and working in schools I hear teachers tell their pupils that they don't get a certain technology or do a certain aspect of computing. We need to remove the acceptableness of teachers being openly helpless at computing in the classroom. Admitting to colleagues that you struggle in an area or need support to move forward is to be encouraged as it can be the beginning of change. If we are negative about the value of the knowledge or say that it is not something we do then our pupils will be negative about it and fail to see its value either.

Why could teacher helplessness be such an important issue in maintaining gender imbalance? I see no evidence to suggest there are more male or female 'helpless' teachers. Let's say 5% of both male and female teachers are helpless. In primary education over 75% of teachers and classroom support assistants are female. So our nominal 5% represents a much higher number of female teachers. All research suggests that positive role models are important in encouraging children to believe that something lies within their domain and computing is no different.

Children are complex and whilst there are no 'boys' or 'girls' programming projects, some pupils of both sexes will enjoy game creation, some connect with the beauty of Maths inside programming, others literacy, music or design and technology. By linking programming to a wide variety of stimuli we demonstrate how it is important for all of us. I have yet to meet a teacher that didn't want their students to achieve more than their own generation was able to achieve, to push the boundaries and remove ceilings. I am confident that when, like me, primary teachers realise how our practice needs to change we will rise to that challenge.

This is an abbreviated version of an article that first appeared on Phil's website: code-it.co.uk. You can read the full piece at goo.gl/NUV8z8

COMPUTING: POTENTIAL KEY TO



COMPUTING FOR ALL



UNLOCK SOCIAL MOBILITY?

Not just a subject for schools in the leafy suburbs. Miles Berry, senior lecturer at The University of Roehampton makes a passionate case to develop strong Computing teaching across all schools.

My grandfather was a coal miner. Both of my parents left school by sixteen. I went to a comprehensive school in the Midlands and went on to read Mathematics at Cambridge. In part at least this achievement was thanks to Dave Pidcock, the Head of Maths at my school, who brought in his Sharp MZ-80k (anybody remember?) so a few of us could spend lunchtimes learning to program.

These are such exciting times. We're the first country in the world to have Computer Science as a curriculum entitlement for all, from age five up. Given the role software plays in all of our lives, it's about time children are learning to write programs as well as use them: even better, that they're learning about the fundamental principles of computation and the processes of computational thinking, as well as the craft skills of coding.

Getting Computing on to the National Curriculum though isn't enough. We've got to make it happen in schools, and it's probably too early to tell how good we are at that. There are some great success stories already, and I think we've certainly hit the ground running... but I hear stories of primary schools where the focus is excessively on Maths and English at the expense of all the other

subjects, of secondary schools without a computing specialist attempting to teach our highly ambitious KS3 curriculum in just two or three half term blocks, and of schools not offering any GCSE CS because they don't have the staff or, and I quote, "it's too hard for our students". As I used to put on far too many school reports: 'has made good progress, although room for improvement remains.'

The picture is a patchy one, but I'm worried about where the patches are. I suspect it's not the grammar schools, academies in nice middle class areas or primary schools in south Farnham that are effectively opting out of providing proper computing. I think it's the challenging schools, where an entitlement to Computing would make the most difference, where it has the least chance to do so. I encourage my trainees to ask hard questions like 'where is the evidence for that?' So where is my evidence for this? I've little evidence there's less provision for Computing in challenging schools, but my colleague, and fellow CAS member, Pete Kemp is on the case and hopes to report shortly. Getting Computing on the curriculum has helped. It really has. Getting Computing taught well, by great teachers able to pass on a passion for the subject, in all schools, helps even more.

CAREER PATHS FOR ALL

Anyone seen The Imitation Game? Who knows who built Colossus, the computer they used to crack the Lorenz cipher at Bletchley Park? Tommy Flowers.

Tommy Flowers was a working class lad, the son of a brick layer, learning engineering through evening classes and an apprenticeship.



Tommy Flowers

We rightly celebrate the achievements of Alan Turing, but I hope not at the expense of working class heroes such as Flowers. We have very little knowledge about social mobility within the computing industry. It's a question few businesses ask or report on yet we need role models from all backgrounds. You see, that's the thing. In Computing, your background doesn't, or at least, shouldn't, really matter. What matters is how good you are at coding, creativity and the attendant problem solving. An entirely meritocratic pathway into making a difference *and* earning a decent salary. What matters now is that youngsters who could become the computer scientists of the future are aware of the possibilities, and get the right encouragement if it's a path they choose.

You don't get to Carnegie Hall just by going to class recorder lessons; school PE isn't enough for an Olympic medal. Julian Sefton-Green's recent report, Mapping Learner Progression into Digital Creativity (goo.gl/13wmae) makes it clear that those who pursue careers in software or digital arts have done plenty outside school. As well as Computing in school, we need Computing after school and in the holidays too. Code Club, Coder Dojo and Young Rewired State are a great start, but let's make sure the uptake fairly reflects the diversity of backgrounds so the gaps get narrow not wider! These initiatives and less formal mentoring need volunteers, able to share their knowledge and pass on some of their enthusiasm. CAS is a community that has brought together teachers, academics and industry experts. Here's an area where the energy of industry specialists could potentially contribute so much. Maybe you could play a part in changing someone's life chances?



DEVELOPING STRATEGIES TO ADDRESS CHALLENGES POSED BY SOCIO-ECONOMIC DEPRIVATION

No-one can doubt the very real and multiple barriers to learning that can exist in some areas with high levels of deprivation. But that is no reason to think strategies cannot be developed to try to address them, argues Paul Powell, who teaches at George Mitchell, a CAS Lead School in Leyton, East London.



IDEAS TO HELP PUPILS NEW TO THE ENGLISH LANGUAGE

As well as those who have a different home language but speak English well, we also have quite a number of students that are new to English. They arrive with little more than “yes”, “no” and “no English / no understand.” Fortunately, in Computing we have many resources to help.

I have often started with Scratch or Kodu. I know Scratch well enough that I don't need to read the blocks and you can change the language. Students love that they can do something in their home language as it takes the strain off for a lesson. When they start to speak a little English the basic language used in such systems actually seems to help – “when the bat touches the cat, the cat die.”

There are other difficulties. I once taught a Year 10 student from Bangladesh who had never touched a computer before and was physically shaking with nerves. Translation isn't always available and doesn't always help. Sometimes they don't know the word in their own language (especially subject specific vocabulary) or are illiterate in their own language.

Early English learners do appreciate the efforts and I am always amazed by how quickly some of them learn and progress.

When I was asked to write something about inclusion from a socio-economic and diversity perspective I was initially a little stumped. My immediate thought was “I don't do anything differently” and perhaps this is the first and most fundamental point. I have found interest from boys and girls from a wide range of backgrounds and I see and share a love of the subject with them. I challenge them, help them and believe in them. I'm sure this is no different to thousands of colleagues up and down the country, but this does not detract from its importance.

My school is in the East End of London and is well into the top quintile for EAL and FSM. The streets that surround us are in the top 10% for deprivation in the country. Last but not least, our Progress 8 is significantly above national average.

Earlier this year, in an ICT mock exam, quite a number of students missed a question because they didn't know what the word ‘efficient’ meant. Many of our students guess the meaning of a word from the class context or find a way not to answer to cover that they don't know. Suddenly it is there in an exam question and you discover that there is a gaping hole in their comprehension and you missed it. At that point all you can do is explain it is an exam and you can't help them.

Every lesson needs me to think about developing language. Even with Year 11 I get them to sound out words, repeat back new and sometimes basic terminology. It might seem childish, but I make a fool of myself while doing it and they seem to forgive me. When it came to the next exam they did better. Language is empowering.

Socioeconomic conditions provide another challenge. Access to computers outside of school can be a difficult and basic skills are not always there. This year about 20% of our Year 7 knew how to save a file. That lack of exposure to technology means we need to take a few steps backwards to start to go forwards. When we key into where they are, it becomes easier and they start to move forward. By the time they are in Year 11 I am sure they will perform at or above national average. I don't have many answers or magic strategies, just a determination to find what works.

DEVELOPING CULTURAL AWARENESS WHILST ENGAGING WITH COMPUTING IDEAS

Culturally Situated Design Tools (csdt.rpi.edu) is a website with activities to introduce Maths and Computing concepts through cultural aspects of predominantly African, African American, Latino and Native American people. The work of Ron Eglash, from Rensselaer Polytechnic Institute, NY, it offers a variety of classroom tools, ranging from simulations to soft-

ware through which students can investigate concepts such as transformational geometry, patterns and algorithms. Explorations of, for example, Native American rug weaving, Afro American cornrow braids and Latin American pyramid structures provide a historical and cultural context for computational thinking activities. Well worth exploring. *Roger Davies*



CROWDSOURCING BOOKS WITH HACK THE CURRICULUM



CAS #include were busy hacking the curriculum again in March, when they crowdsourced another wikibook, this time for A-level Computing. Emma-Ashley Liles reports on a great weekend.

The two days were lead by Peter Kemp. Peter has created previous Computing wikibooks, including our KS3 one in 2015. A Senior Lecturer in Computing Education at Roehampton University, he is particularly interested in social mobility and has been re-researching the subject. We had a wide range of specialists involved, ranging from University lecturers to network engineers, creating high quality content that is now freely accessible.

When teaching Computing for the first time, being less secure, teachers frequently buy textbooks to boost their knowledge. A wikibook could do the same job. More importantly, many students can't afford additional texts to support their studies. With budget cuts, schools don't necessarily have the budgets to stock the vital supporting texts in their own libraries. Many resources available online are not tailored for study at this level. The consequences of this are quite simple. Students from socio-economically disadvantaged backgrounds do less well at Computing, and we believe the lack of good quality free supporting

resources is a contributing factor. Just how much of a contributing factor it is remains to be seen. Peter's research using the national pupil database from the last five years will hopefully be available soon.

We know the books are already helping and some schools are even using the A level book as their primary teaching text. We are also aware of academics from all over the world linking to specific sections of the book to provide support material for their teaching. Most importantly, we see thousands of students using the collaborative section of the A-level book to discuss and share ideas on the AQA pre-release material for the on-screen programming examination.

With this kind of usage in mind, one day in the not so distant future we hope to publish the books in a physical format, allowing school libraries to cheaply offer the supporting material. This will benefit students with poor access to their online format, due to lack of internet access or suitable devices to read it on.



We asked our amazing volunteers (above) why they gave up their weekend and travelled to Birmingham to write our A-level wikibook.

Duncan Maidens who teaches networking at Birmingham City University and provided the venue told us that "With many years teaching networking, the wiki book project is a great way to use my knowledge and demystify the area for A-level students. Knowing this will be available for free maximizes the impact of my effort and makes the whole idea really worth doing." Computer Science teacher Melanie, who has previously contributed to our KS3 wikibook spoke of her "enthusiasm for open platforms and an appreciation of the mission of CAS #include" as the motivation to travel all the way from Devon to take part.

A further wikibook hack weekend was held at King's College London on July 16th. Thirteen authors attended including teachers, academics and industry professionals. Luckily, with the work we have already done, you don't have to give up a weekend to contribute and help #include make Computer Science available to all regardless of their background. The wikibooks are live and we welcome contributions, no matter how large or small. Visit goo.gl/poaWvv. You may even get to see your work in print one day.



AUTISM AND CODING: THE LINKS



SWITCHEDON asked Katharine Childs, East Midlands Code Club Co-ordinator to share some of her insights working with autistic pupils in Primary schools.

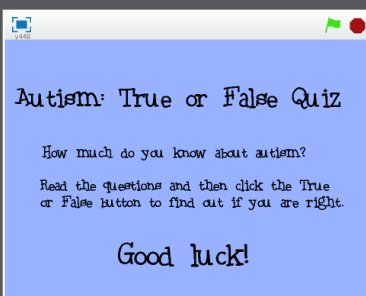
Autism is a lifelong, developmental condition that affects the way people experience the world around them and communicate with others. However, the

autistic spectrum is not a measurable scale of being “a bit” or “a lot” autistic – it’s a rainbow of nuances. If this sounds ambiguous, remember that we humans are complex and unique individuals, and in the same way, autism can affect people in different ways on different days.

In general, when I’ve worked with autistic children in mainstream Primary schools, I’ve seen that their lack of assumptions and their literal understanding of the world around them can be helpful when learning to write code. Conversely, autistic children often find it difficult to abstract detail and model a solution because they want to create everything exactly as it is in real life.

Of course, the goal when learning to code is to be creative and solve problems. So when we set Key Stage 2 homework to make an artefact or item to do with Ancient Greece, I was delighted when a girl on the autistic spectrum wrote a Minotaur’s Maze game in Scratch. For this girl, coding was a platform to express her ideas, just as writing words or drawing pictures can be for other children.

Learning to code brings out other skills too. One of my favourite teaching moments was when a boy who has autism became the class expert and was in demand to help others. The effect that this had on his communication skills and subsequently on his self-esteem was powerful.



Autistic children grow up to be autistic adults, and with only 15% of adults on the spectrum in full-time employment, coding, creating and digital

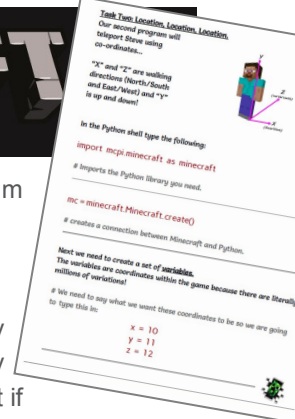
making comprise an obvious career path to consider. How much do you know about autism? Take this quiz written in Scratch to test your knowledge: goo.gl/jaEcvf. The questions are taken from the National Autistic Society’s (goo.gl/yVqJjC) pack for schools and are especially suitable for Key Stage 2.

FACING ‘THE FEAR’ WITH DYSLEXIC PUPILS



Hannah Mills, who is a teacher at Marshfields, a community special school in Peterborough, suggests one way to engage dyslexic students.

You are not alone. Having spoken to many teachers, a lot of us share ‘the fear’ of teaching text-based programming to students with dyslexia. How do we teach them? What activities will allow them to access a language we assume they won’t be able to read or use, given that they struggle so much with every day English? I recently took the plunge and dived in with a class containing two dyslexic students (amongst other needs, as I teach in a SEND school). To my relief it was a great success!



I think the key thing I found was using a medium that they were familiar with and could get on board with from the very beginning... so what did I use? Minecraft Raspberry Pi Edition and Python! My first discovery (this one is probably unsurprising) was that students are more likely to plug away at something until they get it right if it is something they are interested in. I found that those who weren’t that familiar with Minecraft were just as engaged as those heading towards being a pro, because who doesn’t like to play games in their lessons?

We started, as a class, first breaking the code down into each line. Then as we progressed through activities, we gradually put the code used in the last activity together into blocks. This enabled all students, including those who are dyslexic, to make progress towards being able to read and organise longer sequences of code. Talking to the students revealed that they found it easier to type in the code accurately, because they weren’t looking at ‘normal’ words and phrasing; it was just a case of following the line of code carefully when typing it in. The students were also able to debug programs, whether this was a line of code in the wrong place or a typo (deliberate of course!) in a particular line.

Overall feedback was that they enjoyed being able to see the immediate results of their programs in the Minecraft world and that they preferred this activity to block-based programming. They felt it helped them grow in confidence; that actually, not all literacy activities were an uphill, insurmountable struggle. There were some very sad faces when the scheme of work ended. So take the plunge: every Raspberry Pi comes with Minecraft and Python and (with some extra work) you can use much the same code from Python on Windows (see goo.gl/Z2nv0h) and Macs. So jump in and don’t let ‘the fear’ stop you!



TEXT-BASED ADVENTURE GAMES



CAN BOOST MOTIVATION

CAS Master Teacher Matthew Parry, from Stanwick School and Sports College, Derbyshire, outlines an activity combining simple scripting and literacy that even his most reluctant writers have loved.

In my English group I have a number of very reluctant writers. I'm always trying to find ways to get them to put pen to paper or finger to keyboard. Last term we studied 'The Hitchhiker's Guide to the Galaxy' by Douglas Adams, and as we went through the book I let them play on the excellent BBC text game (goo.gl/moG9Cr) that Douglas Adams wrote based on his original radio play.

My students loved the game and one child asked if they could create their own. After some searching and trying out some sites such as Twine (twinery.org) I came across Quest from Textadventures (goo.gl/IQrhzn) and their sister site ActiveLit (activelit.com). I chose to use ActiveLit as it allowed me to set individual logins for my students within a walled garden site for my school.

As an introduction I showed the students one of the games on the site and got them to have a go so that they could see what sort of games they could create. I then got them to plan in their books the game they wanted to create. At first I restricted them to only detailing one character and to designing just three interconnecting areas; this gave them a focus and meant that they would produce a higher level of

detail. I got them to label all the entrances and exits to and from the areas; what objects were to be in each room and how their character was to interact with those objects.

From this, they logged into ActiveLit and started creating their worlds. ActiveLit can run in two modes; we used the simplified version which restricts some of the functionality and also minimises the amount of coding that was required (it was an English lesson after all!). Students created their characters, rooms and objects as they had planned, and added interactions via a simple menu-based scripting wizard. They were also able to add sounds and images to enhance their stories. I then asked the group members to play each other's games and to provide some constructive feedback relating to the plans that had been drawn up. After incorporating the feedback, I let the students expand their games.

Examples of two of the games created can be found at bit.ly/1XbVC00 and bit.ly/1UDddHZ (shown). My students enjoyed making the games and it has been interesting to see where their imaginations have taken them. It has helped with their use of adjectives and verbs as well as their planning of stories (and computer programs!).



The ever growing playlist on CAS TV includes some excellent advice on various aspects of inclusion. Carrie Anne Philbin, director of education at the Raspberry Pi Foundation, discusses some of the issues around gender, inclusion and Computing whilst Dawn Akyürek and Gemma Marsden of King's College School, Madrid, outline some strategies they've used to better engage girls in Computing.

Catherine Elliott also expands on the thinking behind the activities she highlights in this issue of **SWITCHED ON**, whilst discussing Computing and SEN.

Rebecca Franks of The King'swinford School discusses the Pupil Premium and how it can best be spent to raise the attainment of disadvantaged children. Rebecca advocates an approach that is focussed on quality-first teaching and the use of SOLO (Structure of Observed Learning Outcomes), which she explains in the interview. She also discusses other possible uses of Pupil Premium funding, such as provision of internet access at home and to computers.

SUBSCRIBE NOW!

Keep up to date with all new content by clicking the subscribe button. CAS TV can be found at youtube.com/computingatschool.



TEACHING ALGORITHMS TO SEND PUPILS: CLASSROOM IDEAS AND PRACTICAL TIPS

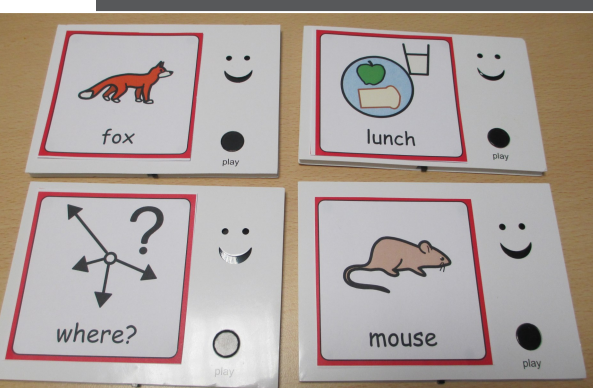
Jeannette Wing's seminal paper states that "*Computational thinking is a fundamental skill for everyone, not just computer scientists.*" Catherine Elliott, SEN Lead for Sheffield City Council's e-Learning Team shares some practical advice on developing these skills with pupils working at the upper P scales.

Written in 2006, Jeannette Wing's paper on Computational Thinking states; "*To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability.*" Computational Thinking is a hugely beneficial skill for the majority of pupils, for example, improving the ability to apply solutions in a variety of contexts, to break down complex problems and be able to order information sequentially. Although there are young people who have not reached a cognitive level advanced enough to make sense of Computational Thinking beyond basic cause and effect, for those working at the upper P scales and above, there are many positives to learning these problem-solving skills which can be applied across the curriculum.

Many Computational Thinking activities can also contribute to the core priorities of these pupils, for example enhancing communication, social skills, numeracy, literacy, life skills and motor skills. This ensures that lessons remain relevant and meaningful for pupils who may never program a computer. Here are a number of simple, accessible activities to teach the Computational Thinking skills of algorithmic thinking and logical reasoning. They are suitable for pupils with special educational needs and disabilities working on the upper P scales and above, but are equally relevant to mainstream KS1 pupils.

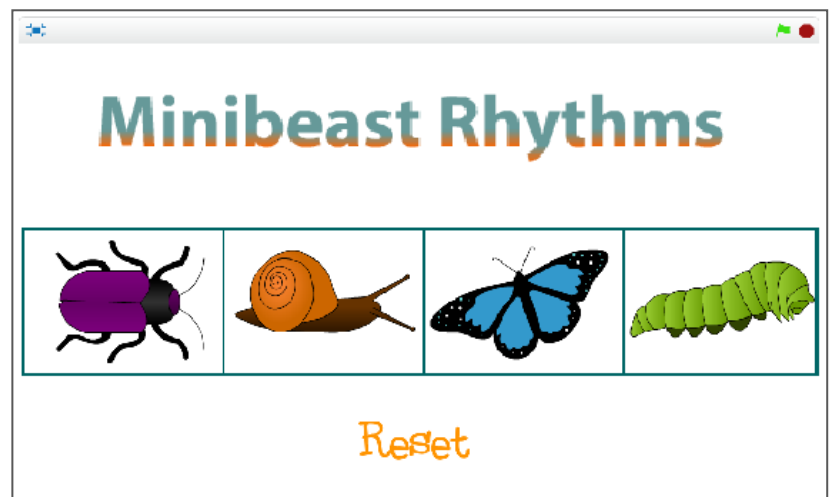
TAKING THE FIRST STEPS WITH STORY SEQUENCING

A first step in algorithmic thinking is putting things in order. There are lots of potential links with literacy, in terms of retelling familiar stories. We can also support a range of needs by using audio and visual support, or 3D objects – pupils can access the task through different senses appropriate to their needs.



Recordable buttons are familiar aids in many schools. You could record key sentences from a familiar story onto recordable buttons, postcards or switches. Stick an appropriate image or symbol onto each button, and ask pupils to put them into the order they appear in the story. Similarly you could read a familiar story or sing a song, and ask pupils to put 3D objects or toy characters in the order they appear.

Using music to teach about sequencing is engaging, and a good way of investigating what happens when you change the order in an algorithm. There are a wide range of recordable buttons available to help with this activity. These allow you to record a short sound clip or section of speech for a pupil to play back. Give each pupil an instrument or sound clip on a recordable button to play. Print out photographs of each pupil and place them in a pile – you may want more than one of each pupil. A volunteer then chooses five cards from the pile and creates a sequence with them on the board. Pupils play their instruments in order according to the photos. Discuss what happens if you change the sequence; i.e. the music will change.



The Minibeast Rhythms activity is a Scratch project. We've shared it at goo.gl/yTTjEX and it can be used to create sequences of clapping rhythms. Choose four minibeasts, and ask the class to clap out the names, e.g. *cat-er-pill-ar-bee-tle-snail-bee-tle* in order. This activity can be extended to start looking at loops: repeating n times, or repeating until the teacher says stop. This is a great way of learning about syllables, and can be done with any number of cross-curricular topics, e.g. transport, colours, shapes or dinosaurs!



Make it relevant Make it sensory Make it fun

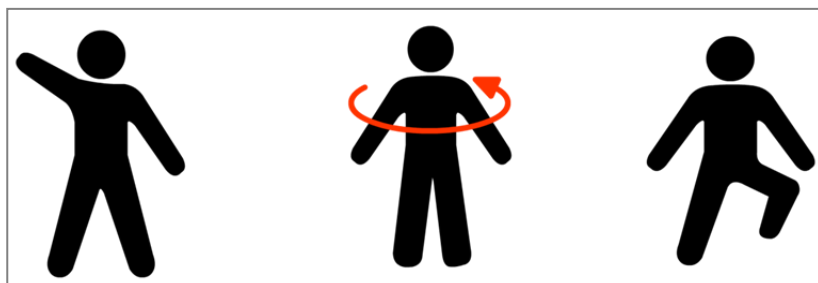


Although pupils working at this level don't need to know what a Bubble Sort is, it is an intuitive way of sorting. It can be used with a variety of objects to teach logical reasoning and algorithmic thinking. In a Bubble Sort, objects or values are sorted (e.g. highest to lowest) by comparing neighbouring objects and swapping position as required until the order is correct. This is best done with each pupil holding a value card or object and swapping places, to encourage movement and communication.

Here are some examples of what you could ask pupils to sort:

- Small amounts of money in coins – supporting numeracy and life skills
- Objects according to weight or size – science links
- Themselves, by height or birthday month
- Textured materials, roughest to smoothest
- Numbers e.g. the answers to simple sums, the amount of items on a card; or perhaps use Top Trump cards (above) and choose a category.

All sorts of activities encourage pupils to move, which can help with co-ordination and motor skills. Also, for many pupils with specific communication and learning difficulties, providing a physical context for an abstract concept also assists with understanding .



TAKING THE IDEAS FURTHER

These are just a few ideas for engaging and including all pupils. For complete lesson ideas see the Teaching SEN page at Barefoot Computing (goo.gl/6cTuu3); you will need to register for a free login. There are a number of activities for pupils with special educational needs looking at algorithms, decomposition, pattern recognition, plus some guidance for teachers. There is also a collection of lesson ideas and useful links on the SEN Computing Wiki at goo.gl/am8tLg along with resources for teaching the OCR Nationals entry-level qualification in Computing.

It is worth visiting the SEN and Disability pages on the CAS #include website (goo.gl/CgTzpB) for more useful links and details of upcoming events. This is also the home of the Revised P Scales for Computing; a document created by a group of teachers and educators around the country which contains a set of statements better reflecting the content of the Computing Programs of Study.

You can create a whole class dance algorithm by sequencing different moves, either using dance cue-cards with images of moves on them (above), or assigning each pupil a different move and then choosing them in different sequences. Again, there is the opportunity here to start to introduce more complex ideas by adding **repeat x times** cards into the algorithm.

Alternatively, create a floor algorithm using laminated cards with symbols and arrows, as shown right. Pupils can work in teams to create a route around the classroom, and decide what movement or task is indicated by different symbols. They then need to explain the rules (the algorithm) to another team. Carol Allen, SEN teacher and consultant in the North East, demonstrated this activity using Ikea placemats at the SEND Conference at the National STEM Centre last year.





DIVERSITY AND INCLUSION IN COMPUTING EDUCATION



The #include Conference took place last June. Alan Harrison, a BCS Scholar teaching at Blessed Thomas Holford Catholic College in Altrincham, reports on an inspiring day.

After lunch Donna Rawling and I ran a wearables workshop. It was a great way to wind down and chat. We had some Electro-Fashion kits from Kitronik as well as a few example pieces already made up. We brought along our collection of sparkly bits, felt and foam flowers as well as needles and extra thread. Our workshopers planned designs on paper and then sewed them onto cotton bags. It took a fair bit of planning to ensure that the traces didn't overlap, that positive never got too close to negative, and that the battery holder was in the 'right' place!



Wearables always look great but can be a bit daunting at first. Students often need help with simple sewing skills, but as long as they can sew in a reasonably

straight line it will be fine. Students of all abilities and ages enjoy the satisfaction of completing a simple sewn circuit; everyone likes a flashing LED!

One of our workshopers works with young offenders and was convinced this kind of activity would be thoroughly engaging. It demands fairly close attention to detail and a fair amount of planning but, once you're past that stage, the sewing begins and getting the LEDs lit is always satisfying. The traces were covered up by adding felt, and the LEDs were used to illuminate flowers, a robot and a rather fetching raspberry too! There was just about time to complete the bags in the two hours allotted. Everyone went away with something to share with their students and colleagues.

Sue Gray

On the day, I discovered what dancing has to do with code, what 11-year-old girls think of programming, and how iPad apps are engaging and stretching the most challenging students. The CAS #include Diversity & Inclusion Conference 2016 was fun and enlightening in equal measure. Among the keynotes that made an impression was that of Katharine Childs, containing valuable insights into autism. Did you know that boys are more likely than girls to be diagnosed as being on the autism spectrum, by a factor of at least 5 to 1? This was followed by one of the day's many firsts: a keynote speech from an 11-year-old, Carrie Anne Philbin's 'protégée' the wonderful Elise, aka @Girls2Geeks, with some great ideas on getting girls into programming and technology. Her answer to my question, "Should we have girls-only code clubs?" was thoughtful and mature, and in a word, "No".



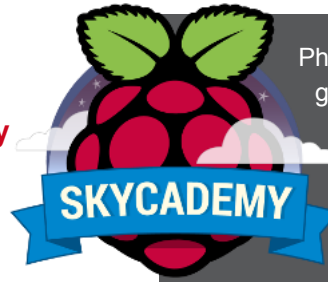
The practical sessions were equally useful, and after coffee I decided to find out from Edge Hill's Dawn Hewitson what a bee's waggle dance had to do with code. As a result, I am now somewhat famous for teaching "Dad Dancing" to a room full of kids and adults. You can see the whole thing here: bit.ly/dancingcode. But what a lovely idea, making the concepts of programming (sequence, iteration, subroutines) accessible to all in a memorably kinaesthetic way!

I simply haven't the space to do the rest of the conference justice, but why not visit my Twitter account, @tech_magpie, for more, or watch highlights on my Storify here: bit.ly/casincludel6. As a quick summary, I'm grateful for insights from Emma-Ashley Liles of 7Digital and @CASinclude, featuring the worrying statistic that less than 300 girls took A-Level Computing last year, and why "hack a hairdryer" was NOT the answer. I gained practical strategies for teaching a 'Haribo bag' of SEND children from the contagiously enthusiastic Hannah Mills, aka @Digitaldivageek, and had a fun session, with CAS Primary Master Teacher Ruth Smith, trying out iPad apps that are great for engaging challenging students. There was just enough time for me to offer my own services to the next conference, share a last coffee and grab some swag before braving the Manchester rain again. Well done all.



ABOVE EAST ANGLIA

Sue Gray, one of the original 24 Skycademy cadets, reports on this fantastic initiative, and the thrills experienced as a result, by students at Fakenham Academy, Norfolk.



In August 2015 the Raspberry Pi Foundation organised their first Skycademy (see sidebar for more details): three days of intense training on how to construct a payload, how to set up tracker boards and pick up signals on radio receivers; as well as how to put together the payload 'train' of balloon, parachute and payload — all joined together with cord. As one of the first cohort of 24 Skycademy 'cadets', I was part of Team Stratus.

Post Skycademy, the first team, Glebe House School, launched in October 2015, but weather and a few tracking issues prevented our launch until later in the year. On 8th May 2016 a small group of students from Fakenham Academy Norfolk (FAN) launched a high-altitude balloon into near space. Students Brandon, Chloe and Charlie had been preparing for the launch for months. They learned how to use a radio receiver to 'listen' to the signals sent from a payload made up of a Raspberry Pi computer plus tracker boards; they also learned a great deal about the weather and weather predictions. Due to our location - about ten miles from the North Norfolk coast - a launch from the school site was not possible. So the FAN group joined

others taking part in the Global Space Balloon Challenge (goo.gl/uJe6NT) under the guidance of Steve Randall from Random Engineering, who supported Skycademy (goo.gl/efhhql) and arranged the launch from his site at Elsworth in Cambridgeshire.

The students were all familiar with the pre-launch preparations of putting the payload together and preparing the 'train'. They were confident handling the equipment and calmly took care of the tracking. The balloon launched at 10:53 and after just five minutes had reached an altitude of 1681 metres; after ten it was at 3282 metres. The balloon burst at 12:13 when it had reached an altitude of 26533 metres.

Sadly, at 12:18 the payload stopped transmitting and the group had to rely on a predicted landing location in order to find it. After a search around the predicted location the payload couldn't be found and the group had to return home. The payload box was decorated with pink duct tape and had a return address label, should it be found. Shortly after arriving home I received a call from Steve Randall to say that he had located the payload! A brilliant end to the day.

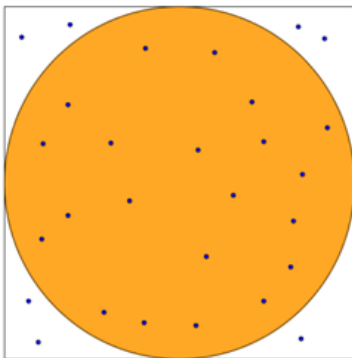
Physical computing gives a real purpose to learning. It gives rise to great project-based experiences, connecting with other disciplines such as sci-

ence, design, engineering and the arts. Two years ago I stumbled across a project that ticked all the boxes: to send a small electronic device (called a tracker) on a helium balloon into the upper atmosphere. During the flight it uses GPS to track its position, and radio to transmit data back to earth. As the balloon rises it expands due to the thinning atmosphere. Somewhere between 20 and 38km it bursts and descends back to earth (under a parachute). Throughout the flight an on-board camera captures images. At above 25km the curvature of the earth is visible, allowing students to view some amazing images.



Thankfully, some seasoned balloonists have created some low-cost hardware for this. The 'Pi in the Sky' board adds this tracker functionality to a Raspberry Pi. Just over a year ago the Raspberry Pi Foundation organised its first CPD event for educators, to get this experience into the hands of students. Over the course of three days our teams built, configured and tested their payload devices, and on the second day they launched five flights across East Anglia. More events were planned for summer and we hope this is just the start. More information at goo.gl/1ZbZvh. *James Robinson*





MATHEMATICAL MUSINGS: USING THE MONTE CARLO METHOD FOR PI

Mark Thornber, who teaches at Durham Johnston Comprehensive School, continues his exploration of ways Computing can be used to enhance students' understanding of mathematical ideas.



In this column I'd like to look at a simple idea that is now used in many different situations. It first came to Stan Ulam when he was working on the atomic bomb project at Los Alamos. His idea was that you could use lots of random guesses to solve a hard problem, and it was one of the early uses for the ENIAC computer. It came to be known as the Monte Carlo method because it came to him while playing cards. We will use it to find the value of π . Of course, this is well known to many decimal places (through the use of computers) but the Monte Carlo method provides one way of finding those digits. It should work with groups familiar with Pythagoras's theorem and circles; usually Year 10 is OK.

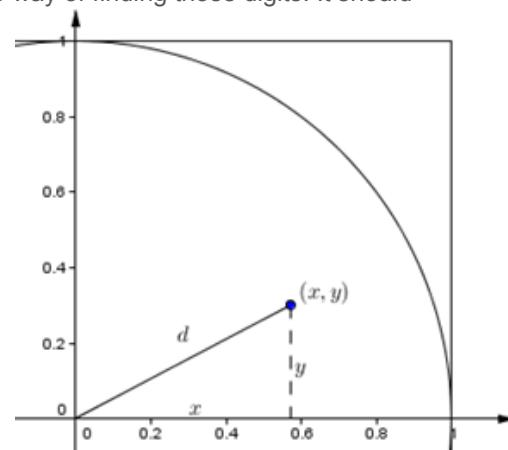
A circle of radius 1 has an area $\pi \times 1^2 = \pi$. We can estimate this area as follows:

- Surround the circle with a square of side 2
- Generate lots of random points in the square
- Count how many are in the circle

The illustration, top left, shows the idea.

We should have
$$\frac{\text{number of points inside}}{\text{total number of points}} \approx \frac{\text{area of circle}}{\text{total area}}$$

This gives
$$\pi \approx 4 \times \frac{\text{number of points inside}}{\text{total number of points}}$$



To make this work nicely I suggest using a circle centred at (0, 0). You can get a random point in the square by generating two random numbers x and y , both between -1 and 1. To check if the point is inside the circle, use Pythagoras to see if the distance to (0, 0) is less than 1 as shown in the diagram above. This means we need $x^2 + y^2 < 1$ for the point to be inside. Here's a Python program:

```
import random
random.seed()
totalPoints = 10000
pointsInside = 0
for i in range(totalPoints):
    x = 2 * random.random() - 1
    y = 2 * random.random() - 1
    if x * x + y * y < 1:
        pointsInside += 1
piValue = 4 * pointsInside/totalPoints
print(piValue)
```

Note there's no need for a square root. That just slows the program!

THE POWER OF COMPUTERS

The accuracy of the estimation of π will improve if you use more points. This is when the power of using a computer program becomes apparent to students. Try generating a million or even more points.

We can take the investigation further. Get your students to add some code to time how long the program takes to perform the calculations. You can time most simple programs with the Python time module. I found one million points took about 1 second to complete on my machine.

```
import time # put at the start
startTime = time.clock()
# code to be timed goes here
endTime = time.clock()
print(endTime - startTime)
```

A further extension might be to challenge the students to predict and then investigate what happens to the time if you generate ten times as many points? What about 100 times as many? Based on their findings, you can ask them to consider whether it is sensible to generate a trillion points.

PyCon UK, the annual gathering of the UK Python community, takes place in Cardiff from 15th to 19th September. Of particular interest to readers of **SWITCHEDON** is the teachers' day, on Friday 16th. Now in its fifth year, it gives you the chance to meet with and learn from developers, and gives them the chance to contribute to the development of resources for the classroom. More details, including information about bursaries are available are on the website: 2016.pyconuk.org. If you're reading this after the event, a full report will follow.

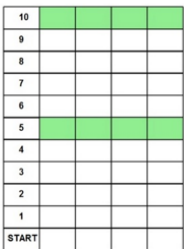
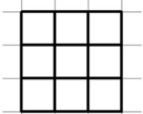
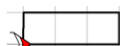

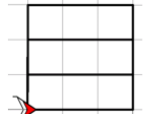
PROGRAMMING AND THE TAO OF



COMPUTATIONAL THINKING

Dave White, a CAS Master Teacher, presents an introductory course exploring the pedagogy of Computational Thinking and programming in Python for teachers at KS2/3.

The ethos is to offer pupils who are beginners in text-based programming a chance to explore cross-curricular topics using a project-based, problem solving approach focusing on the Computational Thinking to create human solutions. Most importantly, it aims to develop and transition to scaffolded program solutions. We start with a Toolbox consisting of three basic motor instructions: *forward*, *left turn* and *right turn*, that drive the familiar sprite/turtle of Scratch, Logo and Python. Using the control structures: *sequence*, *repetition* and *functions*, we develop the toolbox to tackle problems drawing and colouring geometric shapes.

Project 1 Aqado Board and simplified model (decomposition) 3x3 square grid	Sample mission drawing exercises related to Project 1 for the simplified model 3x3 grid			
				
Aqado board	(Decomposition) 3x3 'Model' grid	(a)3x1 Rectangle *	(b)1x3 Rectangle *	(c)Horizontal Rectangles **

Project 1 is an example taken from the problem domain Grids & Board Games. This environment has the advantage of requiring a relatively small basic toolbox to explore it, and produces the reward of a graphical effect when a correct program is run – an important encouragement to pupils. Also this process can be undertaken in any language that implements the sprite/turtle. So what do ISPY and Push-Python have to offer in addition?

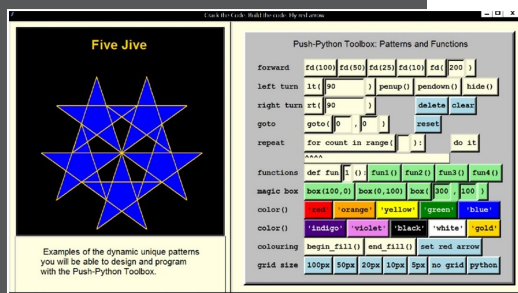
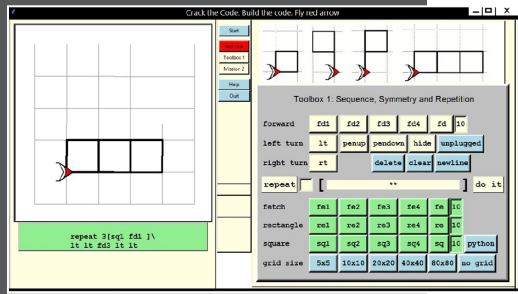
ISPY is a push-button programming technology that allows pupils to build directly, one instruction at a time (with a simultaneous action on the screen), a graphical solution in the text-based language UPL (unplugged programming language) with minimal use of the keyboard. UPL differs from Python and Scratch by having instruction movements recorded as 'paces' rather than pixels. There is a simple mapping to the corresponding instructions in

Scratch and Python. A benefit of push-button technology is that pupils concentrate on the Computational Thinking to construct programs *without* having to deal with difficulties beginners otherwise face when trying to write programs in a text-based language (the cycle of syntax errors, message reporting, editing and re-running). Further, logical/drawing errors are usually immediately apparent and remedied with a push-button delete.

Push-Python is a push-button standalone machine, offering a powerful subset of Python 3, with button formats, screen and program output faithful to an implementation of the turtle in Python 3. It follows on from ISPY, has a much wider capability, and again can be used to take pupils through to constructing programs in Python 3 (using the course projects) with all the advantages of the push-button technology described.

THE PUSH BUTTON ENVIRONMENT: ISPY

There are currently three ISPY Toolboxes, which scaffold the programming process. User-defined instructions (functions) add increased functionality to the Toolbox, and subsequent Toolboxes can be extended by new pupil-defined push-button instructions.



Missions of graded problems related to each project are an integral part of the operation of ISPY, and pupils progress through the problems learning Computational Thinking and practising control programming structures as they go. Finally, ISPY generates the pupil's solution program, which can be saved in a UPL and/or Python version.

The full Course: 'Tao of Computational Thinking in Programming' supported by Google International Awards CS4HS 2015 and CS4HS 2016, will be delivered free to teachers before each term this year and is available for download from ispython.com/tao. The ISPY and Push-Python environments, developed through the CS4HS 2016 award, are also free to download from ispython.com/ispy.



COMPUTING CONCEPTS AND MODELS TO ENRICH THE ENTIRE SCHOOL CURRICULUM

The Turtle System has recently been significantly developed at Oxford University thanks to a project co-funded by the Department for Education. Peter Millican, Professor at Hertford College and author of the system, has written a free book and suite of programs illustrating the value of CS concepts to many subjects.

FREE RESOURCES FOR TEACHERS

It is well known that Computational Thinking is widely applicable, and that programs can be used to illustrate and explore many varied phenomena, especially in the sciences. But most students find it very hard to write such programs for themselves, and even expert teachers typically lack the time to develop more than one or two.

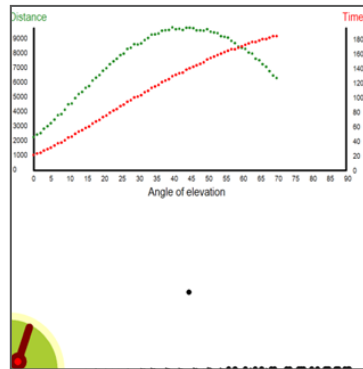
'Computer Science Across the Curriculum' – a book to be distributed free to secondary schools – addresses this problem, and illustrates the implementation of computer models in many different disciplines.

The first chapter explains how to get started with the *Turtle System*, and its basic concepts, after which the second provides a simple introduction (in BASIC, Pascal, and Python) to animation, modelling of motion, and user input by keyboard or mouse. Then follow chapters on Physics, Cellular Automata, Chemistry, Biology, Mathematics (from chaos, recursion and self-similarity to waves), Computer Science (including game algorithms), and Philosophy and the Social Sciences (e.g. models of co-operation and segregation).

The material from the science chapters, illustrated in this article, can easily be incorporated into lessons even without the *Turtle System*, as most programs can run direct from www.turtle.ox.ac.uk/csac. Select one, and it will be loaded into *Turtle Online*; then clicking 'RUN' will execute it in your web browser.

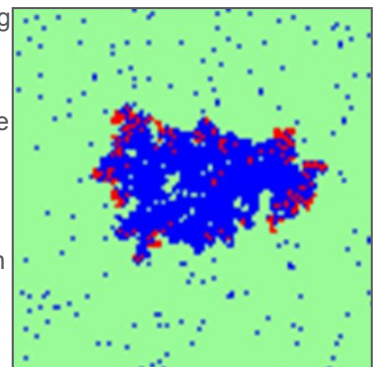
The final chapters — on Philosophy and Social Sciences — are less likely to be relevant to conventional curricula, but are designed to inspire students and teachers to appreciate how computer-based methods open new and exciting possibilities for these disciplines.

Physics provides some obvious topics for modelling. The simple graphical interface of *Turtle System*, based on canvas x and y coordinates, makes it very easy to model motion in two dimensions using a procedure that moves the turtle repeatedly by the appropriate x and y velocities, drawing an animated projectile as it goes. The automated cannon program shown here (with smashed balls littering the ground and one in flight), gradually raises the cannon from 0° towards 90°, building up graphs to show how flight time and distance vary with the initial angle. Students thus gain a deeper, practical appreciation of the relevant theory, from the trigonometric calculation of initial x and y velocities, to the application of gravitational acceleration to the y velocity.



Another more challenging program models firing a rocket into orbit, using real physical values and tracking the rocket's position to the nearest metre each second, as well as its velocity (and acceleration) to the nearest millimetre per second (squared). With control over only the initial thrust, time of thrust, and firing angle, it is surprisingly difficult to achieve orbit, but students are encouraged to develop the program further, taking account of other physical factors such as weight loss (as fuel is burned) and the Earth's rotational velocity. Again, there is scope for entertainment, exploration, and substantial learning beyond the syllabus.

Cellular automata offer lots of interesting possibilities, starting with a simple version of a standard model of epidemics (pictured) which can be used to illustrate the value of inoculation. Then, a simple implementation of the Game of Life – just thirty lines long – aims to motivate learning about binary numbers, Boolean operators and encoding methods, using hexadecimal numbers to store information as coloured pixels.



Another program illustrates Wolfram's theory of one-dimensional automata, generating patterns that are strikingly reminiscent of some found in natural shells (such as *Conus textile* shown).

The chapter on Chemistry uses similar techniques to model diffusion of liquids before moving on to explain and apply simple atomic theory, starting from Brownian motion (which also gives an opportunity to illus-

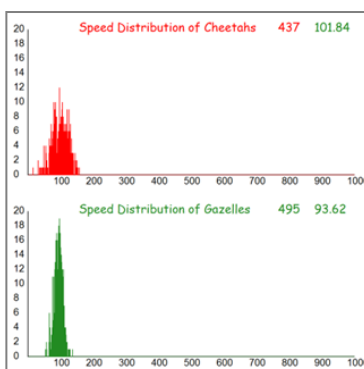
Turtle System is based on Turtle Graphics, an idea invented by Seymour Papert. This sort of programming, and the results it produces, are easy to understand because they are so immediately visual. But the Turtle System discussed here shows that Papert's idea can go well beyond simple graphics, to provide a basis for fascinating and powerful programs that can explore many cross-curricular concepts through a variety of computer models.



CS4FN: FROM FASCINATION TO IMPLEMENTATION

This new book was developed in collaboration with CS4FN, the popular magazine (and website www.cs4fn.org) based at Queen Mary University of London. Most students (and even teachers) are likely to find it hard to translate general ideas into precise algorithms, but these programs illustrate how to do so, and are sufficiently short to be relatively easy to understand, modify and learn from. Topics connected with CS4FN include animal behaviour (e.g. ant trails), animation, artificial intelligence (e.g. playing Nim or Noughts and Crosses), cellular automata (e.g. Game of Life and morphogenesis), chaotic phenomena, disease epidemics, evolutionary models, fractal art, graphics and image encoding, logic, mechanics, Prisoner's Dilemma, and searching.

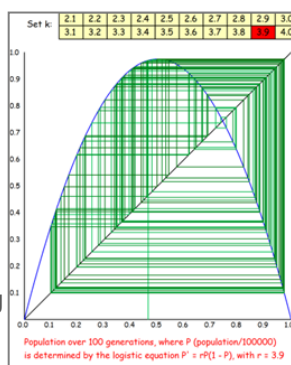
trate more Physics, in the impact of particles). Again independent exploration and interdisciplinary crossover are encouraged, for example by inviting students to combine the diffusion and epidemic models to demonstrate how mobility of infected individuals can radically alter the dynamics of disease spread.



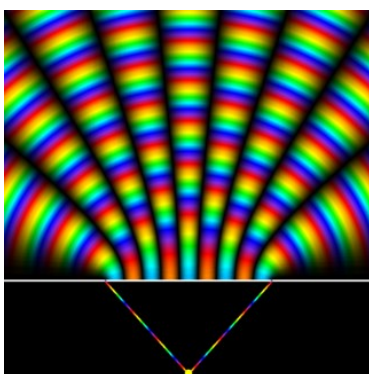
Biology also provides many other excellent examples, from the competitive evolution of ever-faster cheetahs and gazelles, to evolutionary fixing of the sex ratio, to modelling of coordinated movement such as trail-following in ants or flocking of birds. Yet again there is plenty of scope for students to take these ideas further, with the necessary programming techniques explained and illustrated to enable these

models to be developed through, for instance, the introduction of obstacles, rival species, and predators.

Another biological model, of insect population, based on the well-known *logistic equation*, provides an excellent illustration of *chaos*, a form of non-linear dynamics whose wide applicability to many different domains has come to light precisely through the exploration of computer models. For more on this fascinating topic, see the sidebar.



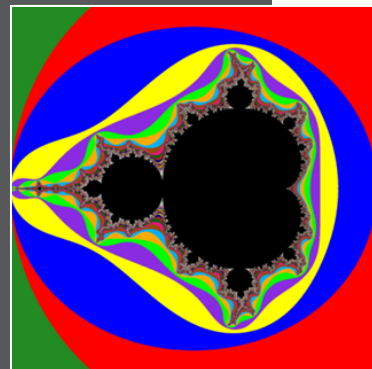
Next comes a chapter on wave interference, including illustrations of



Fourier decomposition (using Hugh Wallis's impressive 'wave superposer' program) and a model of wave patterns within Young's two slit experiment (shown). Though quite advanced, this provides illustrative material for teachers to supplement their classroom explanations, and will help students who may be reading books such as Richard Feynman's *QED* or Brian Cox and Jeff Forshaw's *The Quantum Universe*.

THE MATHEMATICS OF CHAOS

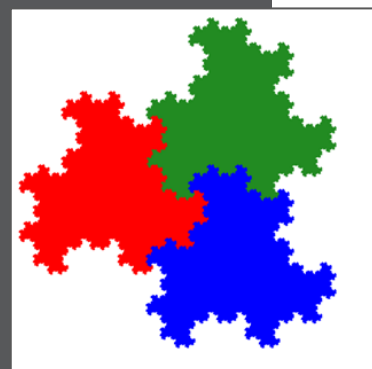
The most celebrated icon of chaos is the Mandelbrot set, and this book explains exactly what that is, providing a simple program that generates a picture of the complete set (shown right), as well as allowing more detailed 'zooming in' to regions within it. The aim here is to give real understanding of the relevant theory by providing genuine implementations – in simple computer language – which can be examined, run and modified by students themselves. Those who have read about chaos, without ever learning exactly what it is, will thus be enabled to dig much deeper.

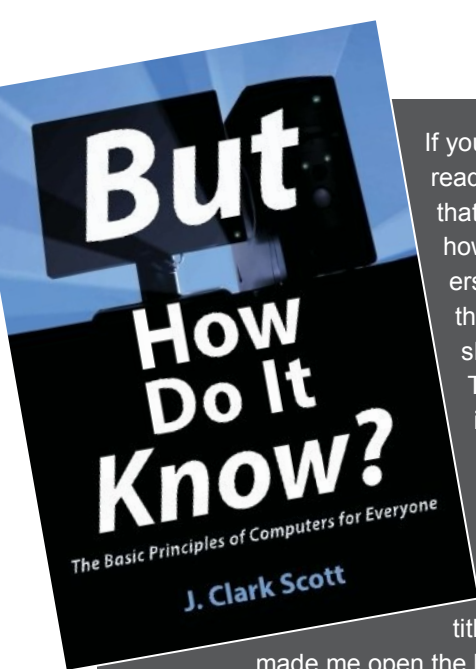


The chaos chapter of *Computer Science Across the Curriculum* then moves through discussion of the well-known Sierpinski triangle, which can be generated in several ways, some of these involving *iterated function systems* similar in principle to the Mandelbrot process. It is fascinating to discover how the simple specification of iterated functions, by re-setting a few parameters, can generate – from what is essentially the very same program – such different patterns as Michael Barnsley's famous fern and the dragon curve (below).



These patterns are known as 'fractals', displaying self-similarity in the complete pattern and their sub-parts, so that, like the Mandelbrot set, in principle they have infinite detail 'all the way down'.





If you only ever read one book that explains how computers work, then this should be it. The title itself has a story of its own. It was to get behind the title that first

made me open the book. You can enjoy discovering that for yourself, but the general idea is to de-mystify the topic of how computers work, something most of us would really appreciate.

In the book, the inner workings of a computer are explained in easy to understand sections. The chapters are very short, which allows the reader to concentrate on absorbing a small section in one session. Many of the chapters are only 2 or 3 pages which makes it very readable. The format also makes it easy to dip in and out of the book, which is pretty much essential for people in teaching jobs. The technical terminology is limited. If your starting point is that a computer has a processor to work things out and some memory where the data and instructions come from, then you'll be fine. What it is not is a manual to explain the workings of an actual computer. The explanations use a simple model computer based on the 'Scott CPU' which is developed, chapter by chapter, to help understand the principles at work.

There must be hundreds of teachers in the UK, leading classes through computer science courses, who don't have the 'proper' undergraduate background. This book can provide all they need (and more) in terms of computer architecture knowledge.

Visit buthowdoitknow.com/ and you'll soon be on the trail to buy the book, watch videos derived from it and read reviews. If you do, expect your confidence levels and understanding to rocket. There is even an online simulator of the processor (shown right), and an Excel version, as described in the book if you want to go a step further.

THE SIMPLE IDEAS THAT MAKE YOUR PROCESSOR TICK

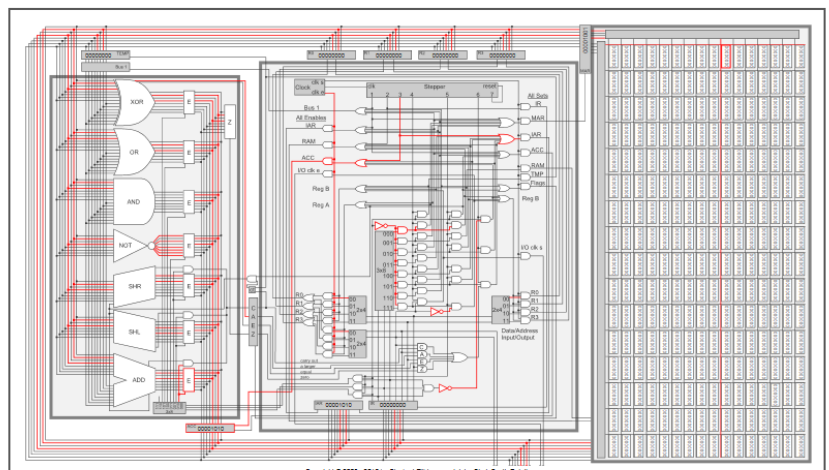
Paul Revell, Head of Computing at The Lakes School, Cumbria, urges all teachers to read a book by John Clark Scott and explore the supporting resources available.



We can't actually see electronics in action, so we need a mental model and this book provides some excellent 'mind pictures' around which an understanding can be built. In particular, many models lead us to an understanding of bits being 'sent' from one component to another and we visualise 'blobs' of data moving along wires. Reading this book soon turns this into a version of live (or not live) wires, allowing a better perception of the speed at which things can happen inside a computer and of their true nature. The wire is made live (or not live) many times per second as opposed to something moving from A to B. It's a bit like turning a light on: we visualise a cold dead wire that is then made live when we flick the switch. It is a small change from the idea of a signal travelling along a wire, but it is disproportionately useful.

Most people reading this will have had experience of teaching students in their mid-teens about logic gates. Almost inevitably the questions "Why do we have to do this?" or "What's the point of this?" will be raised — and, rightly so. If you are left with rather feeble answers, such as "Well they are the basic building blocks of computers" or some equally broad (or vague) response, then read this book; it will add to your own understanding and will give you some easy-to-understand yet absolutely fundamental contexts to pass on to students. It is easy enough to use logic gate simulators to build some replicas of the book's explanations. One of the first things you can do, after reading the first few chapters, is to build a one-bit register with students.

Reading the section about how gates can be joined together to manipulate bit patterns is the core of the book. The way that logical and arithmetic operations are achieved via bit shifts, inversions and the use of adders are all made clear. The answer to the original question (i.e. how does a processor actually know what to do?) is explained by putting together the idea of a decoder and all the gate permutations that can manipulate bits. Essentially, the processor is wired up to perform all the different operations but only the one specified by the decoded input actually 'fires'. This is an excellent book that warrants careful reading.





UNLOCKING THE SECRETS OF WHAT



OUR COMPUTERS CAN DO

In his second instalment reviewing well-respected books on algorithms, John Stout, from King George V College, Southport, takes a closer look at *Algorithms Unlocked*, by Thomas Cormen.

This book is concerned primarily with what computers can do, but the last chapter (see right) covers what they can't do easily or even at all. The first chapter, *What Are Algorithms and Why Should You Care?* does a great job of answering these two questions, introducing topics such as correctness and resource usage (predominantly time). Algorithm analysis starts to make an appearance here by showing how the world's best programmer, using assembly language on the fastest computer, gets beaten by a mediocre programmer using a high-level language on a slow computer with a better algorithm. You'll need a bit of mathematics for this and other chapters, but it's kept fairly simple. You can always skip detail for the summaries.

How to Describe and Evaluate Computer Algorithms introduces simple searching algorithms on arrays and has a nice clear description of big O , theta (Θ) and omega (Ω): notations to classify running times. There's a brief introduction to recursion and it's nice to see a bit of proof (loop invariants) for algorithm correctness here. Chapter 3 extends linear searching to binary search (iterative and recursive) before moving to sorting. Each new algorithm is analysed in more depth than you'll probably ever need to teach, but often brings out nuances that aren't immediately obvious. For example, *selection sort* moves items $\Theta(n)$ times, but *insertion sort* up to $\Theta(n^2)$ times, so if the items are large or on slow storage, the assumption that insertion is better than selection doesn't always apply! *Merge sort* is introduced as a divide-and-conquer algorithm and analysed in depth, as is *Quicksort* which follows. A recap ties it all together. *A Lower Bound for Sorting and How to Beat It* introduces *Counting*

and *Radix* sorts, ways in which we can beat even *Merge sort* and *Quicksort* with certain types of input. *Directed Acyclic Graphs* (DAG) introduces topological sorting and different ways of representing a DAG as an adjacency matrix or list. Chinese cooking is used to introduce PERT charts, critical paths and finding the shortest path in a DAG. *Shortest Paths* explores Dijkstra's algorithm. The importance of data structures is discussed and different implementations using arrays and binary heaps compared. The Bellman-Ford algorithm is a precursor leading to arbitrage opportunities, then the Floyd-Warshall algorithm and its use in dynamic programming!

Algorithms on Strings introduces applications relevant to computational biology. Algorithms include finding the longest common subsequence of two strings, transforming one string to another as 'cheaply' as possible, and finding occurrences of one string within another. The last of these introduces finite automata.

Foundations of Cryptography covers substitution ciphers, one-time pads, and block ciphers: all symmetric key ciphers. By contrast public-key cryptography, the idea underlying internet security, uses different algorithms for encrypting and decrypting; algorithm analysis is essential here. Chapter 9, *Data Compression*, starts with why we want to compress data, why it's possible to do it, and whether we can live with lossy compression (MP3 and JPG compression methods) or need lossless compression (ZIP file compression). The chapter considers Huffman codes (which use binary trees) and finishes with Lempel-Ziv-Welch (LZW) compression.

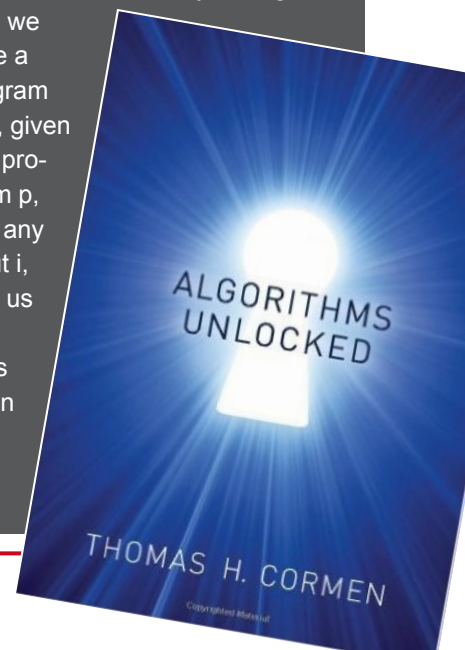
INTRUIGING CHALLENGES WITH HARD PROBLEMS

The last chapter introduces the problem classes P, NP and NP-complete as well as the P=NP problem. P contains the problems we can solve in $O(n^c)$ time (these are tractable even if c is very large); NP contains problems where, given a proposed solution, we can verify that it is a solution in $O(n^c)$ time and NP-complete contains the problems that are in NP and are such that finding an algorithm to solve one of these — in $O(n^c)$ time — means that we can solve all of the NP problems. Also covered are some of the pairs of problems that look very similar but are in different classes, e.g., finding shortest paths in a directed graph is in P but finding the longest path is in NP!

P problems are in NP, but are NP problems in P? This is the P=NP problem. There's a lot here that I skipped reading, but there's a useful table of the different types of problems, as well as a *Perspective* section which discusses instances of problems that are easy and their approximate solutions.

The book finishes with a brief discussion of undecidable problems such as the 'Halting Problem', first proven undecidable by Turing.

Can we write a program that, given any program p , and any input i , tells us if p halts when run with i ?



SIMULATING LOW LEVEL ASSEMBLY LANGUAGE

Low-level programming is often perceived as hard and, as a consequence, was previously reserved for older students, principally studying A-level. More recently, the availability of controlled assessment tasks at GCSE using simulators has provided challenges for a younger age group. Such simulators have a long history. The 'little man computer' has its origins in an analogy first developed at MIT. The little man fetches, decodes and executes each low-level instruction in turn. Using a very restricted instruction set helps bring home to students how the high-level constructs they take for granted are implemented at the level of the machine.

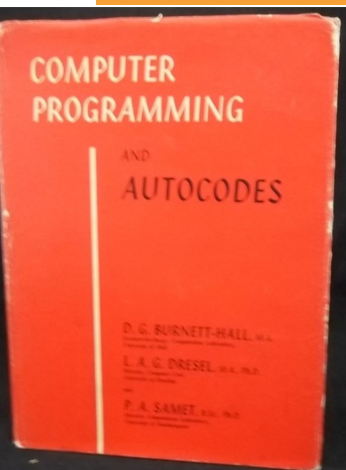


An entertaining and simple introductory simulator suitable for younger KS3 children is available on the CAS Community at resources/1383. It uses a numeric (decimal) instruction set. Using a sample, or writing their own short sequence, students can then

watch the little man (shown) explain each step, or give him the instructions to run the program stored in memory. If low-level programs are new to you, use it to tackle the second exercise below. *Roger Davies*

A COUPLE OF CLASS EXERCISES

1. Show how to translate for and repeat loops into unstructured code.
2. Show how to translate conditional and iterative statements into your favourite assembly language.



For details of early programming languages for British computers, see D. G. Burnett-Hall et al, *Computer Programming and Autocodes*, English Universities Press, 1964. Although long since out of print, second-hand copies do become available on Amazon and other outlets.

GOTO CONSIDERED BENEFICIAL IN MEMORY MACHINES



Using 'goto' in code is widely regarded as bad practice. Greg Michaelson, Professor of Computer Science at Herriot-Watt University, suggests some potential benefits.

I think that, in essence, computers are memory machines. Memories are associations of addresses and contents, and, at the lowest level, both are bit sequences. The beauty of bit sequences is that they have no inherent meanings but can be made to behave as if they represent arbitrary entities: hence the universal power of computers.

A most important concept is that *a bit sequence in memory can represent the address of another bit sequence in memory*. Thus, to function at all, computers depend fundamentally on *indirection*: the ability to get bit sequences from memory, or put them into memory, using other bit sequences as addresses.

In a von Neumann CPU, a running program is sequenced by the fetch/execute cycle, through repeated indirection on the program counter (PC) holding the address of the next instruction. Blocks of instructions are held in consecutive locations in memory and, after each instruction is executed, the PC may be automatically incremented to give the address of the next instruction.

However, this implicit linear sequencing can be changed by a *branch* instruction which includes an explicit address to be placed in the PC. Branches are either *unconditional*, or *conditional* on some program state, for example flags set after operations.

High-level programming languages abstract away from the underlying memory machine. We tend to focus on variables as the central abstraction from address/contents associations, but some of the earliest abstractions were:

- *labels* to abstract from addresses in code;
- *goto* statements to abstract from unconditional branches, by specifying labels identifying where the program is to continue;
- *if* statements to abstract from conditional branch instructions following operations.

Typically, there were also:

- *for* statements, abstracting from conditional branch iterations over groups of instructions;

However, most other control flows had to be crafted explicitly out of *labels*, *ifs* and *gotos*.

As computer use exploded through the 1950s and 1960s, there was a growing expectation that they could be used to tackle increasingly complex problems. But, as programs grew in size and functionality, it took longer and longer to develop them and it became increasingly hard to maintain them, especially as there were no well-established standard methodologies for systematic software development.

In 1968, in a highly influential paper, the pioneering Dutch Computer Scientist Edsger Dijkstra argued that a key factor in this alleged *software*



crisis was the unconstrained use of *gotos*, observing that complex chains of branches led to programs that were hard for anyone other than the original programmer to understand. Instead, Dijkstra urged the use of *structured programming*, based on the core building blocks of sequences, conditions and iterations, with *gotos* entirely banished.

Thereafter, structured programming was steadily adopted both in industry and for teaching, and a new generation of imperative languages either heavily restricted (C/C++/C#) or entirely lacked (Java, Python) *gotos*. Thus, today it is most unlikely that anyone learning to program will encounter the *goto* statement.

Nonetheless, to run on memory machines, high-level *goto*-free programs must still be compiled into lower level forms with explicit branches.

Let's explore this in a bit more detail, using my favourite language BASIC. Suppose that, for program structuring, we only have the unconditional **GOTO <line>** and the conditional **IF <condition> GOTO <line>**. Suppose also that A, B, C etc. are line numbers.

We can realise **IF...THEN...ELSE...** as:

IF <condition> THEN <statements1> ELSE <statements2> END IF	A IF NOT <condition> GOTO D B <statements1> C GOTO E D <statements2> E
--	---

For example:

IF X>Y THEN PRINT X ELSE PRINT Y END IF	5 IF X<=Y THEN 20 10 PRINT X 15 GOTO 25 20 PRINT Y 25 ...
---	---

Similarly we can realise **IF...THEN...** as:

IF <condition> THEN <statements1> END IF	A IF NOT <condition> GOTO C B <statements1> C ...
--	--

For example:

IF X>0 THEN PRINT "POSITIVE" END IF	5 IF X<=0 GOTO 15 10 PRINT "POSITIVE" 15
---	--

Finally, we can realise a **WHILE** loop as:

WHILE <condition> DO <statements> END WHILE	A IF NOT <condition> GOTO D B <statements> C GOTO A D ...
---	--

For example:

SET X TO 0 WHILE N>0 DO SET X TO X+N SET N TO N-1 END WHILE	5 LET X = 0 10 IF N<= GOTO 30 15 LET X = X+N 20 LET N = N-1 25 GOTO 10 30 ...
---	---

I think that understanding the computer as a memory machine gives integrative insights into the deep connections between total immersion in a favourite multi-user game and billions of transistors turning on and off very very quickly. Each year I show our second year undergraduate students these equivalences, using C and ARM assembler in our case. And, each year, several have commented that they now better understand the implications of their programming choices, and how helpful it would have been to have seen this much earlier.

THE CURSE OF SPAGHETTI CODE

Programs built from unconstrained *gotos* are justly derided as *spaghetti code*. To see how spaghetti clouds clarity, let's compare structured to unstructured code for a slightly larger example using our translation templates. Here's a binary search:

```

SET L TO 1
SET R TO N
WHILE L<=R DO
    SET M TO (L+R)/2
    IF V=A[M] THEN
        BREAK
    ELSE
        IF V<A[M] THEN
            SET R TO M-1
        ELSE
            SET L TO M+1
        END IF
    END IF
END WHILE

```

And here's the equivalent unstructured BASIC:

```

5 LET L = 1
10 LET R = N
15 IF L>R GOTO 60
20 LET M = (L+R)/2
25 IF V<>A[M] GOTO 35
30 GOTO 60
35 IF V>=A[M] GOTO 50
40 LET R = M-1
45 GOTO 55
50 LET L = M+1
55 GOTO 15
60 ...

```

Even knowing what the BASIC is supposed to do, it's much harder to follow the control flow and work out what's going on. Note the redundant branch to line 55 on line 45. At the machine code level, an optimising compiler would spot this and patch the equivalent of line 45 to branch directly to line 15.

A PAUSE FOR THOUGHT

Pause for a moment to consider a classroom forty years ago. There were no computers, no internet, no projectors, not even photocopiers. Pause for a moment to consider how easy the 'establishment' might dismiss those promoting the virtues of the personal computer — still in its expensive infancy.

Fast forward: the new millennium witnesses an unprecedented boom in school IT. If, like me, you winced at the prevalence of educational software that replicated 19th Century 'skill and drill' methodology, pause for a moment to consider the words of Seymour Papert, written forty years ago.

Contrasting such 'instructionism' with his own 'constructionism', he notes: "One might say the computer is being used to program the child. In my vision, the child programs the computer, and in doing so, both acquires a sense of mastery over a piece of powerful technology and establishes an intense contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building."

Seymour Papert passed away on July 31st, aged 88. His ideas were ahead of his time and have heavily influenced a new generation of Computing educationalists, including, for example, Mitch Resnick, creator of Scratch. In today's short term, target-driven environment, let your class pause for a moment. In Papert's words, give them "time to think, to dream, to gaze, to get a new idea and try it and drop it or persist, time to talk, to see other people's work and their reaction to yours." Let them pause for thought. In so doing, draw confidence, knowing your practice is following in the footsteps of an educational giant. *Roger Davies*

UK BEBRAS CHALLENGE

In the second week of November, all schools, Primary and Secondary, can again enter their students into the UK Bebras Challenge. Each year the number of students taking part has nearly doubled; we are hoping for over 100,000 this year! One reason for the continued growth is that we prioritise participation for all and work hard to help schools administer the challenge without hassle. Students can take part during their normal Computing lessons in Bebras week and we work with schools to accommodate them if they have a technical problem or practical issues such as running a two-week timetable.



The Bebras Challenge does not require any programming ability but instead asks students to solve engaging Computational Thinking problems. We are also building a legacy of problems that can be attempted by students or assigned by teachers at any time during the year. There are PDF booklets published with full answers and an explanation of the Computer Science behind the problems. All of this information can be found by visiting bebras.uk.

As well as ensuring schools have plenty of certificates to give out, last year we invited some of the most amazingly talented students up to Hertford College and the Computer Science Department at Oxford University for a final round and celebration. We intend to run this over two weekends, at the end of January and the beginning of February 2017, so that we can invite more students. With ever-increasing numbers, we rely on our sponsors to help us keep UK Bebras free to schools. We are immensely grateful to the Raspberry Pi Foundation and ARM Holdings who have seen us through the early years of UK Bebras, and for the hard work of Professor Peter Millican and his team at Oxford. It has been a pleasure to have Google join us this year, encouraging us to be more ambitious than ever and generously providing funding to realise those ambitions.

TAKING THE NEXT STEP WITH THE KESTREL CHALLENGE

Having identified so many talented UK students, how can we help them make progress? We are adding a *Where Next* page to the Bebras website with links to some amazing resources and introducing a new pilot competition (The Kestrel Challenge) in March. The top 10% of students in the four oldest age categories in the Bebras challenge will be eligible. Students will be asked to code solutions to Computational Thinking problems. Again the challenges will be archived in a self-marking format for all schools to access later. We will also provide a set of specimen challenges and tutorials to help students make the transition. The challenges will consist of some easier programming problems at the start (see the example below where Blockly commands are supplied) followed by more extended problems that can be solved using any programming language. Each challenge will take an hour to complete. More details will follow shortly.

Chris Roffey

Logo Maker: A new country has been founded from five small, friendly countries. A program is required that can be used to create its new logo. Using only the blocks supplied, write a program that draws the shape shown. You may alter the variable values but you must keep the line length equal to 100 units.

