COMPUTING AT SCHOOL
EDUCATE · ENGAGE · ENCOURAGE
In collaboration with BCS, The Chartered Institute for IT

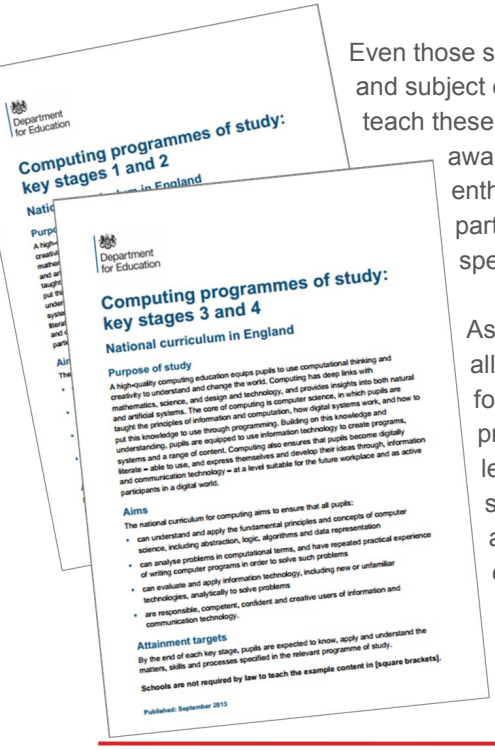
SWITCHED ON

COMPUTING AT SCHOOL NEWSLETTER AUTUMN 2014

COMPUTING, PROGRAMMING AND PEDAGOGY

So now it's for real. No school can afford to ignore the challenges set by the introduction of the new Computing curriculum. No one should underestimate those challenges either. Even the most confident teachers are sailing into uncharted waters.

Some schools may still be grappling with the ***why***. Why are we teaching Computing? Others may well have embraced the need for change: the need to equip a new generation with the tools to make sense of a rapidly changing technological future. But that still begs the question of ***what***. What do we need to teach children to develop their ability to 'think computationally'? There is no shortage of resources, but welding them into a coherent scheme of work will take time. Schools will need to be flexible and willing to adapt as pupils become exposed to the core ideas of computing at an increasingly earlier age.


Department for Education

Computing programmes of study: key stages 1 and 2

Department for Education

Computing programmes of study: key stages 3 and 4

National curriculum in England

Purpose of study

A high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world. Computing has deep links with mathematics, science, and design and technology, and provides insights into both natural and artificial systems. The core of computing is computer science, in which pupils are taught the principles of information and computation, how digital systems work, and how to put this knowledge to use through programming. Building on this knowledge and understanding, pupils are equipped to use information technology to create programs, systems and a range of content. Computing also ensures that pupils become digitally literate – able to use, and express themselves and develop their ideas through, information and communication technology – at a level suitable for the future workplace and as active participants in a digital world.

Aims

The national curriculum for computing aims to ensure that all pupils:

- can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation
- can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems
- can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems
- are responsible, competent, confident and creative users of information and communication technology.

Attainment targets

By the end of each key stage, pupils are expected to know, apply and understand the matters, skills and processes specified in the relevant programme of study.

Schools are not required by law to teach the example content in [square brackets].

Published: September 2013

Even those schools with established schemes of work and subject expertise will be asking ***how***. How do we teach these concepts effectively? Teachers are well aware that learning requires effective pedagogy, enthusiasm and deep subject knowledge on their part. But there is another element too: subject specific pedagogy.

As more teachers embrace Computing, we will all have ideas to share. No-one has a blueprint for what works best. CAS hopes to bring fellow professionals together in the Network of Excellence, to share ideas and develop that subject specific pedagogy. Good teaching is, above all, a collegiate activity. The next few pages outline various ways that CAS can help you, from planning tools to CPD, and from communities of practice to accreditation for your efforts. But above all, they carry the insights of colleagues starting to 'walk the walk'.

INSIDE THIS ISSUE

The "Computing At School" group (CAS) is a membership association in partnership with BCS, The Chartered Institute for IT and supported by Microsoft, Google and others. It aims to support and promote the teaching of computing in UK schools.

bcs Academy of Computing

ISSN: 2050 -1277 (online) 2050 -1269 (print)

FACING UP TO THE COMPUTING CHALLENGE

The introduction of Computing is the most momentous change teachers of ICT have ever faced. Far more than a rebranding exercise, it marks the start of a significantly different focus to what is taught, how it is taught and why. Secretaries of State for Education will come and go but Computing is here to stay, regardless of reshuffles or the outcome of the election next May. Computing heralds a recognition by those in power of a new educational imperative.

Putting coding on the curriculum is often the simplistic media portrayal of this important change. CAS has long argued that there is a lot more to Computing than coding. At its heart it is about building the capacity to solve problems through computational thinking. One of the best practical expressions of a child's capability is to engage in computer programming. But how do you teach children to develop the underlying thinking skills? This is the key challenge.

When a subject is new and teachers are unsure of what to teach, it is tempting to reach for ready made resources, of which there are many. But copying has never been a particularly effective method of learning - for pupils or teachers. Of course, pre-prepared materials can save a lot of preparation time, but 'adapt rather than adopt' might be a prudent mantra.

The most important part of developing our new curriculum will come from reflecting on our experiences (good and bad) and discussing them with supportive colleagues. That is what CAS is all about; a genuine community of practice, founded on a willingness to try things. Just like the children in our charge, we will learn most by 'doing', sharing and thinking about what works best. *Roger Davies*

DIY COURSE TO HELP TEACHERS TEACH TEACHERS COMPUTING

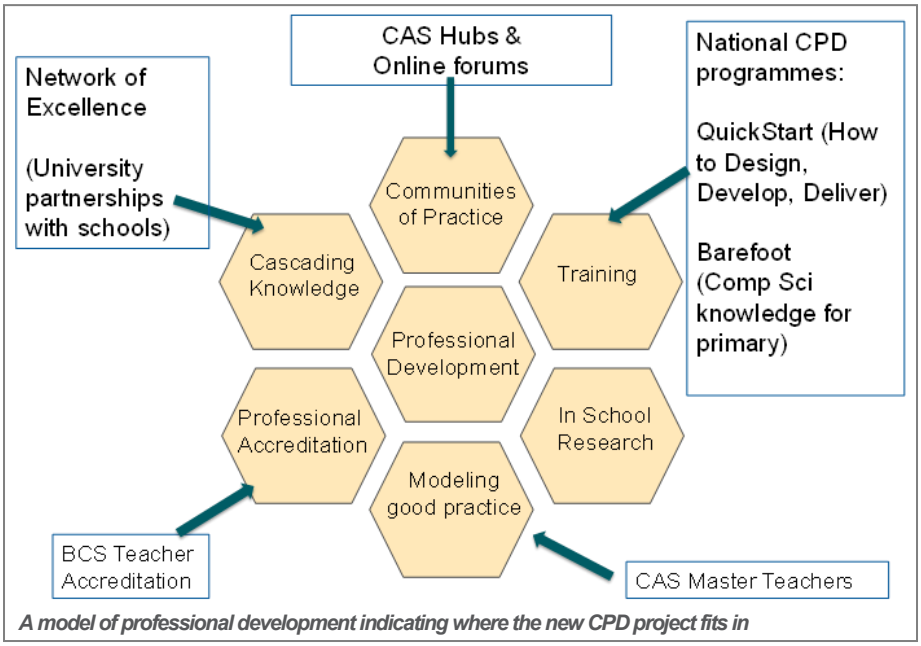
Figuring out how to teach Computing well will be a long and interesting journey. Bill Mitchell, Director of Education at the BCS Academy of Computing outlines a major new initiative to help schools.

The CAS QuickStart Computing project, funded jointly by the DfE and Microsoft to the tune of half a million pounds, will be a huge boost to help schools get going on their journey. The project will generate a free shrink-wrapped Computing CPD course designed for a volunteer teacher to run for their colleagues. It will involve high quality written materials to support face to face sessions and an interactive platform to guide teachers through consolidation exercises. The course will be freely available to all teachers to download, and we aim to give over forty thousand hard copies to schools.

The CPD course will help teachers answer some of the most basic questions about how to teach computing including: how to plan and prepare my first ever term of Computing; how to provide the right progression for my students; how to assess progression against the curriculum; where to get authoritative guidance on the new curriculum; how to audit existing ICT teaching resources to see what still maps to the new Computing Curriculum and how to evaluate other people's teaching resources to see which parts are useful. Teachers who know how to deal with those questions should be able to confidently face their students, as well as the most challenging of senior managers or even possibly an Ofsted inspection.

The intention is that the CPD course could be run by a volunteer teacher through a local CAS Hub, or through a Lead School in the Network of Teaching Excellence in Computer Science (NoE), or could be run in-school with just a few immediate colleagues from the same department. The intention is for it to be run by someone with reasonable experience of teaching Computing, but who is not necessarily an expert. The key aim is helping teachers design, develop and deliver their own new Computing curriculum. A 'beta' version of the course will be published online in the early autumn,

There's no shortage of excellent resources, but how do teachers make sense of it all?



A model of professional development indicating where the new CPD project fits in

while the fully polished final version will be launched at the BETT show in January 2015.

Why do we need such a course? After all there are so many excellent classroom resources available online, both free and commercial. For example, the CAS Barefoot Computing project (see p8) is funded by the DfE and is putting together exemplification resources for primary schools showing how to teach computer science concepts within a cross-curricular environment. Code Club, CS4FN, CS Unplugged and Code.org are just some of the other excellent places to go for teaching resources that cover both secondary and primary education.

Possibly the biggest problem a teacher new to Computing faces is how to make sense of all those riches and assemble them into compelling and inspirational schemes of work and accompanying lesson plans that will cover the school year. Especially when that teacher is teaching themselves Computing at the same time as their students. For example, one of the most common misconception of someone new to Computing is that it's all about coding.

Coding can be hugely exciting and very creative when done right, but it's only one part of the curriculum. Coding becomes valuable educationally when it is used as a mechanism to apply computational principles and concepts to solve exciting and challenging problems. Indeed coding can be seen as the ultimate test of whether someone does understand how to apply computational principles to solve challenging problems. But you'll never code well unless you understand those computational principles and concepts in the first place.

In fact unplugged style activities are some of the best resources for teaching computing concepts. That means they are based on children working together in groups on physical activities using only paper and pencils (and quite often cardboard tubes and bits of string) to explore how computing concepts work in practice, with no electronic device anywhere to be seen. Of course as a CAS member you probably already know that.

So yes there's lots of *stuff* teachers can use in class, but it's not clear how to put all that *stuff* together in a way that means students will properly progress through the curriculum. There are lots of schemes of work on the Web too, but how do you evaluate them to see if they are any good and how do you adapt them to incorporate a different set of progression pathways through the curriculum? There is therefore a definite need for a CPD course that helps teachers answer the basic questions we listed at the start. We hope the CAS QuickStart Computing material can address that need and be taken up by hubs and schools.

QuickStart Computing

The new computing curriculum is quite possibly the biggest educational bootstrapping exercise for at least a generation. Although there are a steadily growing number of CAS Master Teachers who are busily supporting colleagues with the help of many universities around the country, there simply aren't enough of them to meet the demand from teachers over the coming year.

Our motto has always been 'there is no them only us', so it is very appropriate that CAS is creating a CPD course that is designed to be run by teachers for teachers as a do-it-yourself course.

The new course will cover the CAS Primary Guidance document, the CAS Secondary Guidance Document, the CAS Progression Pathways, example outlines of schemes of work, and some of the most well-known classroom resources currently in use, so that teachers have concrete examples to work from. The course will also come with explanations of some of the key concepts in the curriculum. Videos that explain key concepts such as computational thinking will be provided to back up these explanations. Software packages that are useful exemplars of teaching resources and that are free to use will also be included.

The physical copy of the course will include copies of these other documents and a DVD containing videos and software. Wherever possible the course will signpost or include well known examples that teachers already use, and explain how those can be mapped against the guidance documents and progression pathway document accompanying the course.

VIEWS FROM TEACHERS IN THE PILOT PHASE

Matt Wimpenny-Smith is the ICT and Computing Co-ordinator at Headington Prep School in Oxford. He explains; "I saw the course advertised on the CAS website. It has been excellent and I would recommend any ICT teacher to apply. Having completed the course I now feel more confident to deliver the new primary computing curriculum. It was run through SharePoint, making it easy to access. Being online also enabled sharing of ideas with others. The first part was a reflection on my CPD. For the second part I create a computer program using PHP, SQL and Java. For the final section I wrote a case study about how I had been teaching Scratch and Kodu to year 5."

Linda Rowe, the 2i/c and Key Stage ICT & Computing Co-ordinator at Cramlington Learning Village added, "Being part of the pilot was a very rewarding experience. It was undoubtedly hard work, as I extended my CPD in order to build my knowledge and skills, in preparation for teaching the new curriculum. I really valued exploring the 3 elements of the certificate. I am confident that it will have a positive impact on my teaching and benefit my students. This has already been evident through my pedagogical research task and through introducing Python programming and other elements of Computer Science into my lessons. I have been teaching for a while now so it was important to me to have accreditation for Computer Science, which was not part of my original teaching Degree. This has enabled me to continue to develop both personally and professionally. I would highly recommend it."

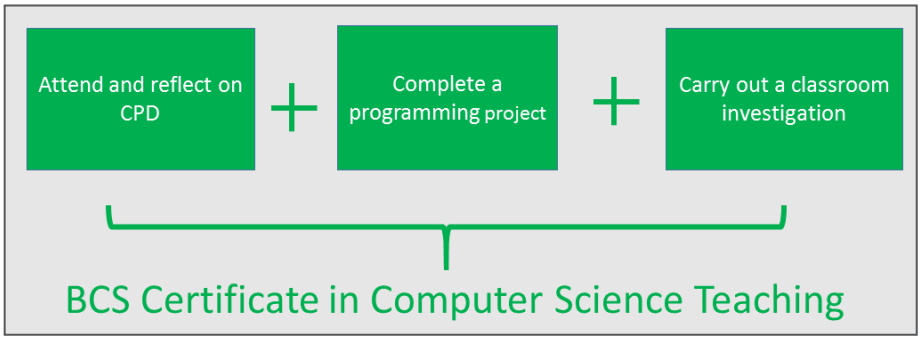
RECOGNISING TEACHERS WHO ARE STEPPING UP TO COMPUTING

The BCS Certificate in Computer Science Teaching is a new form of accreditation for Computing teachers. Following a very successful pilot phase, Sue Sentence invites you to enrol.

The BCS Certificate in Computer Science Teaching enables teachers to demonstrate their teaching competence in the Computer Science elements of the new Computing curriculum. Teaching Computing requires a good understanding of Computer Science as it is taught in school and the development of appropriate pedagogical skills. The Certificate provides professional recognition for this. Working towards the certificate will help you if you do not have formal teaching qualifications in Computing, but need to demonstrate your competence in order to progress and gain recognition. It will also help you to consolidate and extend your existing skills and support your work in the classroom. There are two versions of the certificate, one specifically for primary teachers and one for secondary teachers.

The cost to enrol on either the primary or secondary teacher versions of the BCS Certificate is **£300** + VAT. Further information about how to enrol can be found at computingatschool.org.uk/certificate. This is a roll-on, roll-off programme and you need to complete your evidence within one year from the day you enrol.

The certificate has been devised and implemented by Computing At School and is accredited by BCS, based on your feedback as teachers. We are very grateful to all the teachers who have completed the pilot and have received the first BCS Certificate in Computer Science Teaching awards. We will be publishing their names on the CAS website.



To be awarded the BCS Certificate, you will need three type of evidence:

- Evidence that you have taken responsibility for your professional development by attending, and reflecting on, appropriate training (and other developmental) events that meet your own needs. Training in how to deliver the new Computing curriculum is being offered locally by your CAS Master Teacher.
- Evidence that you have appropriate knowledge and programming skills beyond the level you are teaching. This should be a working project. Examples from the pilot included a Scratch system to record house points, a tool to help KS2 remember their homework tasks, a quiz on computer hardware and an adventure game demonstrate file handling and arrays.
- Evidence that you are able to use a range of pedagogical strategies for computer science and are able to investigate their effectiveness in your classroom and with your pupils. Examples include investigating approaches to teaching sorting, paired programming as a method of learning to code and supporting numeracy in Computing lessons.

LOCAL HUBS HELP TO FOSTER A SENSE OF COMMON PURPOSE

With the new curriculum there is no need to go it alone. Leader of Computing at St Mary's Catholic College on the Wirral, Victoria Tatler urges you to get along to your nearest CAS hub.

Since entering the profession three years ago as an ICT specialist, I have taken every opportunity to get to the forefront of the changing curriculum. I enrolled in university, up skilled, but was mostly inspired through sharing good practice and learning new skills in a CAS Hub. As Computing was new, I recognised the need to source a support network and so launched the Wirral Hub. After contacting CAS, I invited teachers from primary and secondary education and representatives from local universities. The first meeting established what the group wanted from the Hub. I found Twitter invaluable in creating awareness and advertising each event and for on-going communication.

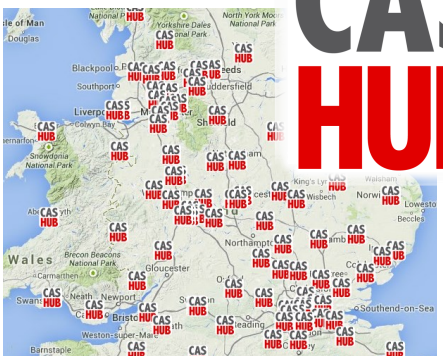
Although the Hub is inclusive, it is important to break off into working groups to ensure each short meeting has the most impact. The Hub is a place where members can learn new skills, identify and fill gaps in their knowledge and share tried and tested ideas. CAS Hubs seek to develop confidence and we try to commit to providing members with simple and effective ideas which can be used in the classroom the very next day.

An additional benefit I have found from leading the Hub was establishing a vital link between St Mary's and our partner primary schools. This link between primary and secondary teachers will ensure students are engaged in Computing from a young age, entering secondary education with a knowledge of Computing. The members drive the success of the Hub. I have been amazed with the positive energy at each meeting, and the willingness to share fantastic resources.

Developing a network and being a Hub Leader has also provided personal career development. I'm now keen to offer further support as a CAS Master Teacher. I would recommend joining or leading a Hub, regardless of where you are on your Computing journey. Hubs are a powerful tool to enthuse, excite and remove the fear that comes with delivering such a dynamic new subject. My advice? Don't go it alone, get involved.

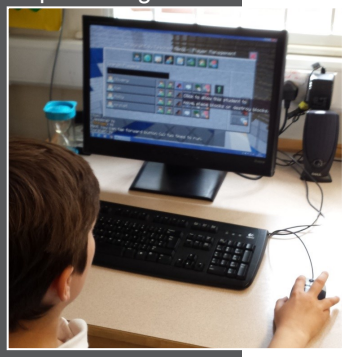
HUB GENERATES HOST OF ACTIVITY IN NE SCOTLAND

There is lots of CAS activity north of the border! Claire Griffiths, CAS Hub leader for North-East Scotland, organised a Computing themed CPD day for Teachers in Moray. Activities included an opportunity to set up a Raspberry Pi. There were discussions on how the Pi could be used in the classroom alongside other innovative hardware and software. Later in the day there was an opportunity to see satellite tracking with a venetian blind/Arduino set-up and a demonstration of the T-Exchange's home-made 3-D printer. T-Exchange Moray provided expert advice and Highlands and Enterprise allowed use of meeting rooms at no cost. The catering was provided by contributions from local firms and two volunteers. In March over a hundred young people attended a CAS Coding Fun Factory. The all-day event took place as part of the Moray College Science Festival Families Day. There was enthusiastic interest in a new code club for 9-11 years, which joined the existing Coderdojo Moray for 12-17 year olds. A Lego Mindstorms themed Coding Club meeting followed. Work from the two monthly Saturday clubs will be displayed in September at a STEM event in Forres.



FINDING INSPIRATION

As a secondary teacher, my lessons tend to be specialised. It was therefore inspiring to see how primary staff approach Computing, often combining subjects in projects. Craven Hub was buzzing at Settle Primary when Sarah Entwistle and her Year 6 Digital Leaders introduced MinecraftEdu. Sarah explained how it has developed pupil literacy, describing scenes for stories in detail. She could drop students into worlds to explore different eras in history or set a survival task promoting team work. They can build cities, create music or develop Maths and Geography by working out location co-ordinates.



MinecraftEdu is proprietary. It comes with a range of prebuilt worlds. Students can work collaboratively and communicate with the teacher. Sarah sets objectives that are rewarded with materials to use to carry out further tasks. Whilst not a programming environment, it gives children an introduction to the software and could lead to programming with python in Picraft at a later date. I think the most valuable aspect of MinecraftEdu was the cool factor. Children are keen to use it, learn new techniques, talk about successes and improve their skills. What can you say? What a fantastic way to get students switched on to Computing! *Emma Partridge*

A FRAMEWORK TO DEVELOP COMPUTATIONAL THINKING

Computational thinking sits at the heart of the new programme of study for Computing: “A high quality computing education equips pupils to use computational thinking and creativity to understand and change the world”. By contrast there is an interpretation, led by the popular media, implying that the new computing curriculum focuses on ‘coding’. This gives a misleading message, especially to teachers who are new to the subject. Computer science is a discipline with its own body of knowledge that can equip pupils to become independent learners, evaluators and potentially designers of new technologies. In studying computer science, pupils gain not only knowledge but also a unique way of thinking about problems: computational thinking. Unfortunately there isn't an agreed definition, list of concepts or defined set of techniques to support teachers in understanding what computational thinking is, how to integrate it into lessons, evidence it, and link it to assessment. Without such guidance, computational thinking is in danger of becoming simply a ‘buzz’ word.

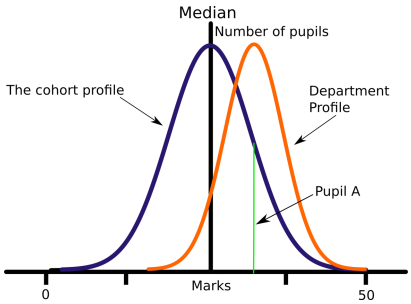
In the last issue of SWITCHEDON, the Progression Pathways framework was offered as one interpretation of the breadth and depth of the programme of study. It has now been updated to include computational thinking and provides a framework for developing computational thinking in the classroom. Beside each learning statement (for all strands of computing) there is now a list of one or more computational thinking concepts that might be possible to evidence during learning. Although useful, on their own these statements don't explain how computational thinking can be embedded into the classroom. A four-step framework also helps teachers by setting out practical ways to understand and introduce the ideas together with a case study. It can support the planning of activities to develop computational thinking skills and to assess progress in doing so. The updated version is on the CAS Community (resources/2324). *Mark Dorling*

WHAT IMPACT DOES A TEACHER MAKE?

The advent of a new curriculum subject brings a unique research opportunity. Ian Lynch outlines plans for a baseline testing project and urges you to sign up.

A new subject starting is a rare opportunity to find out what children know before formal teaching starts on a national scale and then follow this up afterwards to see what difference teaching has made. The computing baseline testing project is a joint initiative from NAACE and TLM, supported by members of Mirandanet and CAS to try and establish a nationally representative sample of data based on what children understand related to the new KS3 National Curriculum. Asking a range of questions to children in Y6, 7, 8 and 9 will give an idea of how much basic knowledge they have before formal teaching begins. We will also be able to see the pre-teaching differences simply due to age.

The test design has certain aims. First of all it must be simple and low cost to administer. We have no funding and want it to be a free service. A multiple choice quiz is used with a variety of questions including simple knowledge and more complex process type questions that need analysis. Without a taught background the quiz will be hard but we decided it is best to make it difficult. If it was too easy there will be little scope to see improvement. We also want to be able to test pretty well anyone as it is an opportunity to see what is known up through the age range. There is no reason for teachers not to take it, for example. We will keep all individual and school results confidential to the identified assessor that owns the account. That person decides who can access the results. We will not give individual results to government agencies as we don't want accountability measures to distort things. This is a community project intended for internal purposes to inform teaching and learning. It is an example where a professional initiative can replace large scale political direction. The technologies we all believe to be important tools in education are an essential agent for support.



We will be able to place individuals within their cohort. For example if they are at the tenth percentile consistently in Y7, Y8, and Y9 it is very likely they will be in Y10 and Y11, suggesting a grade A / A*. If the percentile shows decline the candidate is making slower progress than the cohort, and vice versa. We can also judge whether the cohort has gained or lost overall. Similarly for a school population so you will know if your students progress faster or slower on average than others. Since we are looking at progress, there is little to be gained in disclosing test questions before the test. We hope the professionalism of teachers and absence of accountability constraints will make this a diagnostic tool. We won't know how it works in practice until we try it. If anyone wants to take part just sign up using the signup link at awards.theingots.org (top right). The more pupils that take the on-line test the better quality the data will be so the more the better. It's free to take part. All we need are contact details to send you instructions.

JOIN THE INTERNATIONAL BEBRAS COMPUTATIONAL THINKING CONTEST

Last year the UK joined 30 other countries around the world in offering students the opportunity to enter the Bebras international informatics competition. Organiser, Chris Roffey, looks forward to an even bigger entry this year.

The Bebras Competition (bebras.org) is an annual international contest based around computational thinking and computer science (called Informatics in many countries). Now in its 10th year, children aged from 6 to 18 can enter through their schools. The competition takes place in the second week of November and is run online. It takes less than an hour and the results are available as soon as the competition week is over. Teachers can build activities around it and it represents a great opportunity to link problem-solving puzzle-like activities to principles of computer science.

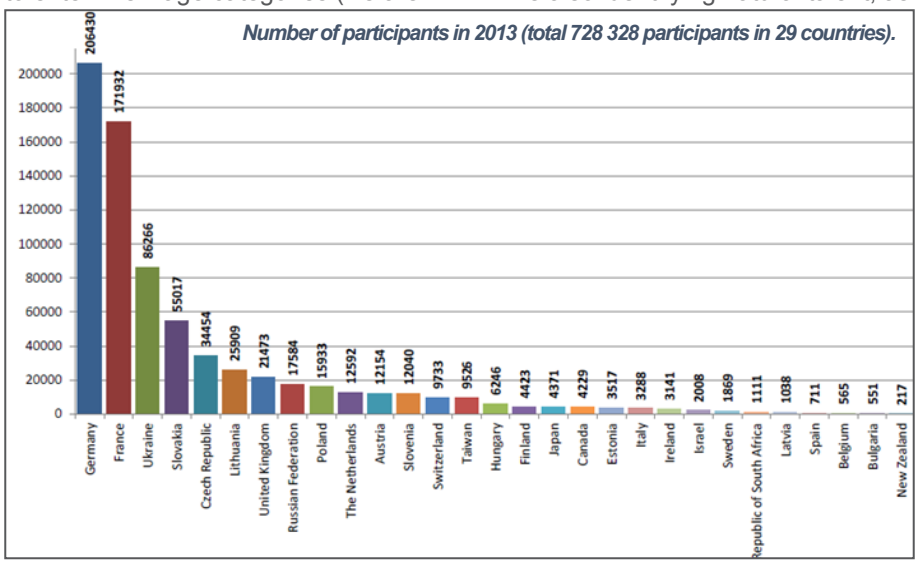
Last year saw the first UK Beaver Contest (www.beaver-comp.org.uk) for students in England, Scotland, Wales and British International Schools (NI students joined the All Ireland Competition). Over 21,000 students were entered by 210 schools which was just amazing for the first year. This year we want to increase that number so we would love more of you to join in! Registration will be open in September and the competition will be in the week of 10th-14th November. This year you will be able to enter in six age categories (we are adding the two new younger age groups for the first time): Years 2-3; Years 4-5; Years 6-7; Years 8-9; Years 10-11 and Years 12-13.

The Bebras competition questions are all in the area of computational thinking skills and include puzzles and problems involving algorithmic and logical thinking in their solution. You can see a sample in the column on the right. Participating in this competition is not just about the week in which it is entered online; there is also the opportunity to work with students on similar problems before the competition and to explore the computer science in the questions after the competition. This year we plan to share how the questions are related to computer science when we publish the answers.

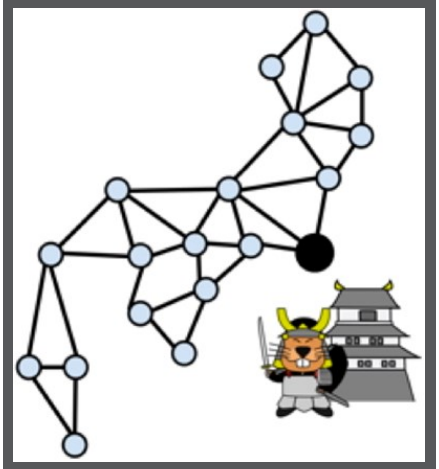
Feedback from schools and statistical analysis showed that our pilot year was very successful, with some fantastic feedback about the students' enthusiasm in all age groups and genders. It was an accessible competition that raised enthusiasm for computer science in schools across the UK while also identifying natural talent, as demonstrated by the Hall of Fame on the website. The competition proved to be gender neutral in appeal.

This year Sue Sentance and I spent a week with the Bebras team in Lithuania editing questions to ensure that the tasks set are suitable for a wide range of students around the world.

We believe this competition is innovative and exciting. It enhances the understanding of and education in computing for children aged between 6 and 18. It promotes STEM thinking skills directly and aims to increase enthusiasm for Computer Science and it identifies and fosters excellence. Moreover, it provides a very useful measure for schools of the spread of their pupil's reasoning and problem solving abilities.



LIGHTING THE BEACONS



A long time ago, the beaver samurai of Japan built a network of beacons. When an emergency occurred, the beacons were lit to warn the whole country. When a beacon is lit, it would take a minute before the neighbouring beacons saw the signal fire. They then light the fire in their own beacons. After another minute, the next neighbours will see the signal fire and light their beacons. This continues until all the beacons are lit.

One day the beacon of the headquarters was lit (the large black circle on the map). ***How many minutes did it take until all the beacons were lit?***

BAREFOOT VOLUNTEERS BUILD SUPPORT IN PRIMARY SCHOOLS

The Barefoot Computing team have been hard at work over the summer readying their primary resources along with a national network of Barefoot Volunteers to help primary schools make the transition from ICT.

Barefoot's Computing resources are designed to enable teachers to develop their own computer science subject knowledge by firstly explaining key computer science concepts linked to the computing programme of study and then offering suggestions for classroom activities to teach these concepts. When writing our resources we've kept the computing novice firmly in mind and have enjoyed creating animations to accompany explanations to convey the concepts as clearly as we can! We've also related our teaching activities to the wider curriculum providing a context for pupils' developing understanding of computer science. All our resources can be accessed through www.barefootcas.org.uk.

To spread the Barefoot Computing message, a number of events have taken place including the launch in London on July the 11th where delegates heard from a range of speakers including David Brown (HMI National Lead for Computing) and Miles Berry (Principal Lecturer and subject leader for Computing Education at Roehampton University). Delegates also tried a range of the Barefoot Computing resources as members of the Barefoot team tested a selection of unplugged activities with the audience. In the afternoon, Scratch Junior was officially launched with a live link with Mitch Resnick of MIT. To spread the message further, in the week after, Barefoot headed north with a conference in Manchester. The Barefoot team also enjoyed working with primary teachers at the CAS Conference for Teachers at The University of Birmingham. The focus of all these events was to provide teachers with the opportunity to explore Barefoot resources and signpost them to the website for access to even more!

To ensure as many schools as possible benefit from the Barefoot resources, over the past few months, a network of volunteers from computing industries and universities have been recruited and trained to deliver free Barefoot workshops in primary schools across the country. To help organise this vast network of computing enthusiasts, Barefoot has established links with a number of partners, including the South West Grid for Learning, Code Club Pro, Connect Education & Business and many others. As schools now start to host Barefoot workshops and use our resources, we are establishing Barefoot Computing Communities in partnership with the CAS Network of Excellence. These self-help groups will reach out to teachers in the local community, encouraging them to participate in the development and sharing of best practice in computing teaching and learning. Through a culture of collaboration and reflection these groups will sustain long-term improvements on pupil outcomes.

Overall it has been an exciting initial few months for the Barefoot Computing Project, and we're looking forward to helping primary teachers in schools across the country make a successful, fun and enjoyable start to delivering the Computing curriculum. Please visit www.barefootcas.org.uk to find out more about the project and make sure you register to be kept updated on future developments.

The Barefoot Team

WHAT IS IT ALL ABOUT?

The Barefoot Computing project is a new initiative geared towards helping primary school teachers get ready for the computer science element of the new computing curriculum. The project is funded by the Department for Education and is busy developing high-quality, practical cross-curricular computer science resources and workshops to support primary school teachers in England. The Barefoot Team is made up of Pat Hughes, John Woollard, Miles Berry, Jon Chippindall, Jane Waite and Zoe Ross.

MANAGING CURRICULUM CHANGE WITH A ROLLING PROGRAMME

With so many possible resources available, where do you start? Andrew Shields, an SLE with a focus on Computing, currently teaches a reception class in Leicestershire. He suggests ways to introduce the new curriculum across the primary school.

There has been a lot of talk regarding what teachers are going to do once September arrives. Which scheme of work should you choose? There are some well regarded commercial options. Alternatively, the well thought out and resourced Somerset materials (bit.ly/1d1P2OK) are free to use as long as you acknowledge the source. You could scour various websites, blogs and twitter to come up with your own. Any of these will give you a good starting point from which to grow a scheme of work that suits your school's specific needs.

Each school will approach September from a different starting point. Some of you will have dabbled in Scratch, Hopscotch, algorithms and be quite adept at using Bee-Bots. Others will not. Children will generally come at this with an open mind and absorb all they can making quick progress, whereas the adults may be approaching it with a small amount of fear and dread. Whatever your starting point, there will be members of staff in your school who will be at your level and those who are not. There will be children who have done a lot, perhaps at home or in a club, and those who have not. We need to cater for all eventualities.

The new computing curriculum is a big step change, especially the coding elements. There are aspects of what we are asked to teach year 6 they will not have come across before. That may also hold true for the year fives and fours. If you think about elements of the new Spelling, Punctuation and Grammar syllabus, teachers have had to back-fill to make up for gaps in prior learning before moving forward. You may need to do the same with Computing. In this first year you could have the whole of key stage two working on year 3 units for computing, in the second year have years 4, 5 and 6 work on year 4 units and so on over the next few years. You may speed up this process if faster progress is made. Key stage 1 may also benefit from the same kind of approach.

A lot of what is taught in key stage 1 and to some extent key stage 2, can be done away from a computer. There are a number of free resources available. Phil Bagge's website www.code-it.co.uk and csunplugged.org are well worth a look if you want a fun activity that can be done on the carpet, in the hall or in the playground. Many schools are also making use of 'Digital Leaders'; capable children able to teach others, both children and adults. As adults, we cannot be experts in all areas of the curriculum, sometimes we need to be brave and put our trust in the children.

Membership of the CAS Community, attendance at CAS CPD and Hub meetings are ways to develop our own knowledge. I have found them very useful and continue to look for new offerings. CAS Master Teachers are dotted around the country organising training opportunities to get people up to speed and these will increase. Don't feel everything must be in place on day one. Each week there is another organisation, website or resource made available. Use the next few years finding out where you, your staff and children are up to, what gaps there are and what training needs organising. You may need to upgrade some kit but probably don't need to make a big grab for cash. See what you have first and how it fits your needs. Let's all move forward with our eyes, ears and minds open and make the most of this new opportunity.

DIGITAL BADGES SUPPORT PROGRESSION PATHWAYS

In the last issue of SwitchedOn I wrote about using digital badges at Snowsfields Primary School to support the assessment of programming. We have since worked with Makwav.es to map the whole of the Computing Programme of Study. You can view the first set of badges at bit.ly/WhazBz. I'm using the Computing Progression Pathways and each badge is matched to the coloured levels of each strand.

To achieve a badge pupils must complete tasks assigned to Badge Missions and upload their evidence. The badges are hosted on Makewav.es but are Mozilla Open Badges, so can be transferred to their Open Badge Backpacks once they reach 13.

The badges utilise a mixture of 'plugged' and 'unplugged' resources with opportunities to develop their own independent learning. Not everything needed for a mission needs to be taught. Children will take time to develop their own understanding and pupils will earn them at a pace suited to their ability.

As I am a primary practitioner I have only begun to map badges for the Primary Phase (Pink - Purple). Links could easily be made to 'levels' beyond these. Feedback will be invaluable for further development to accredit the learning that takes place. If anybody who would like to develop KS3 badges please contact me as your knowledge will be crucial in meeting the needs and demands of secondary teachers who may plan to use them. *Matt Rogers*

SCAFFOLDING TO DEVELOP PROBLEM SOLVING CAPABILITY

Freedom to explore is essential for learning. But how do you support and challenge a mixed ability class engaged in a variety of projects? Andy Colley, from St Mary's Catholic High School, Astley offers some suggestions.

My most successful lessons have all shared a couple of attributes. The first is 'freedom within a framework'; a task with clear success criteria where pupils have a measure of control over the output which helps engage them. The second is the tricky concept of 'challenge and support'. When coding this is especially difficult. There could be 25 plus different projects with individual requirements. Whilst I have become quite adept (I think) at supporting the lower end, it is more difficult to ensure flexibility to allow more able students to go deeper and further without relying on teacher support.

So my problem is threefold, I need to plan so that :

- Learners attempt to solve the task independently without relying on me because they can't be bothered to think.
- Those who can go further are stretched and challenged without having to wait for me to be freed up to set them the extension task.
- I have time to help those who are genuinely stuck.

I make heavy use of video tutorials. They allow pupils to access my teaching at the relevant point in the lesson. This takes care of bullet one - have they accessed the video? No? Then do that before calling me over. However, the problem with video is embedding the challenge. Are they working problems out for themselves or just copying from the screen? So, to scaffold the challenge and help to solve bullet 2, here's a four step method I'm adopting when posing coding problems in Scratch.

Step 1: Show them the pseudo-code
Firstly, learners get the pseudo-code and try to create their code blocks based on it. If they can't then they move to step 2.

Step 2: Code blocks not joined together
Step 2 shows learners a screenshot of all the blocks needed for this piece of code. However, they are not arranged, so the pupil has to work out how to combine them.

Step 3: Blocks joined together
Learners should be encouraged to move away from steps 3 & 4 as soon as they feel able (or a bit before) as it involves less thinking. However, the support does need to be there, particularly for trickier bits of code.

Step 4:The video
Usually used when learning how to use Scratch, this shows pupils where the blocks can be found in the program and gives them a voice over to explain the context of the problem.

By engaging with the pseudo-code first they are also preparing themselves for more text based coding later. I've just started to adopt this approach with some classes, and I'm finding that they really enjoy the challenge and try to work with as little support as possible. It's going to take a considerable effort to retro-fit this into my existing Scratch units, but I feel that it will be well worth the work.

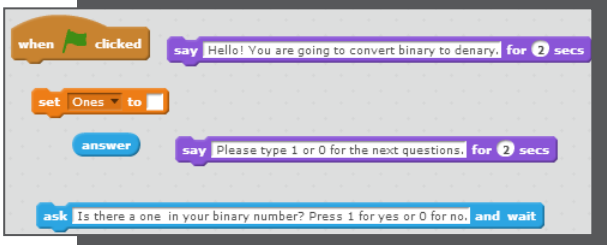
A FOUR STEP SCAFFOLDING EXEMPLAR USING SCRATCH

The example below is a small exemplar of the approach outlined in the main feature. We are trying to code a binary to denary converter. The program needs to ask the user to input a binary number one digit at a time. The snippet introduces the program and the question to input the first digit.

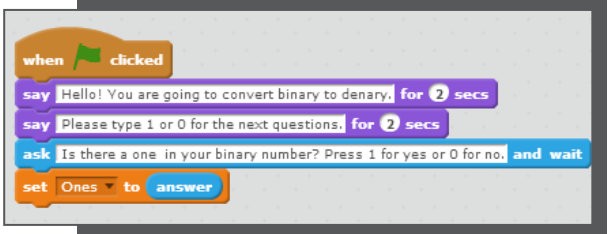
Step1: The stages in pseudocode

```
when the green flag is clicked
say "Hello! You are going to convert binary to denary."
wait 2 seconds
say "Please type 1 or 0 for the next few questions."
say "Is there a one in your binary number? Press 1 for yes or 0 for no"
wait for answer
set variable 'Ones' to answer
```

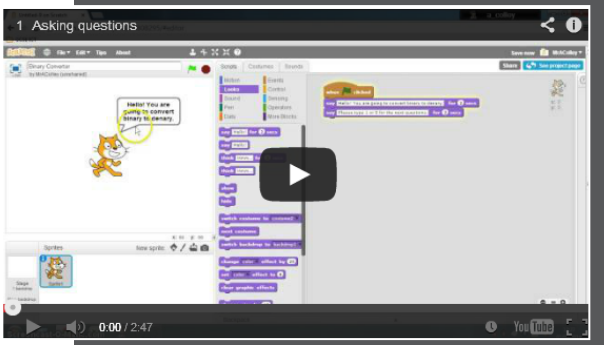
Step 2: Providing deconstructed blocks



Step 3: The constructed blocks



Step 4: Finally, the help video – made with Screencast-O-Matic



This approach has been directly inspired by the great work done by Phil Bagge and the Computing department at Bourne Grammar including Marc Scott. More ideas on my website: mrcolley.com

WHAT FACTORS CONTRIBUTE TO AN OUTSTANDING CODING LESSON?

Phil Bagge, a Computing Inspector/Advisor in the Hampshire Inspection & Advisory Service and a CAS Primary Master Teacher, shares some insights into what makes a good programming lesson, and points out some pitfalls to avoid.

Debugging is the skill of finding programming faults. It involves the understanding that making mistakes and learning from them is a normal part of programming. Implicit in this is that it is not the teacher's job to fix pupils code. In fact to do so invites dependence on the teacher. Bearing this in mind, a good programming lesson will see the teacher establishing ways for pupils to fix problems themselves. In the early stages, with block based programming, such as Scratch, this might involve encouraging them to compare their code with the teacher's example or a friend's. It may involve encouraging pupils to try and identify the specific problem area by reading the blocks and seeing if the narrative makes sense. With text-based programming languages it may involve checking through a list of common types of fault before asking for more help. The first job of the teacher is to help pupils fix errors themselves. In my experience many teachers find this very difficult and need to retrain themselves. It is better to stand back as a teacher in a programming lesson than undermine pupils independence through intervening.

All programmers accept that their first attempts at an algorithm or program may be incorrect. In fact it is the struggle to develop a working solution that is intriguing and intellectually satisfying. Whilst it is good for pupils to solve many challenges in a lesson, teachers shouldn't be afraid to leave threads of challenge unsolved. These extra hooks can intrigue some pupils and inspire them to work on solutions in their own time. A good programming lesson will have appropriate levels of struggle and wherever possible unfinished elements or unanswered questions to intrigue.

When I started primary programming, I read an excellent post by CAS # Include about how some male teachers would refuse to fix boys' programming problems but when faced with girls struggling with the same issues would provide solutions. I read this with a certain amount of smugness, sure that I would never succumb to such inherent sexism. However, at a programming club that afternoon I spotted myself challenging a group of boys to find the solution themselves and then giving a group of girls, not six metres away, a solution to a similar issue. I was horrified that these 1950's attitudes still prevailed in my practice. To train myself out of doing this, the first step was to recognise I was doing it. I now have more girls attending coding clubs and everyone is equally challenged and allowed to struggle.

Maths interaction needs to be anticipated and prepared for in an excellent programming lesson. Programming is a wonderful way to prove that maths concepts work in the real world, be it decimal fractions, angles, coordinates or many other areas. For example if using decimal fractions to reduce the speed of movement of a sprite in Scratch a pre-drawn line split into tenths can help illustrate the point. Pupils don't need to understand every aspect of maths to use it but a good programming lesson will add another layer to their understanding.

Programming is wonderfully open-ended. I have found over the last few years that pupils are capable of coming up with better solutions than mine on many occasions. I am not the fount of all knowledge; I am a fellow traveller on the road towards understanding. In a good programming lesson pupils know that their contributions are valued. A solution can be changed or added to by their contribution and celebrated by their teacher.

Finally, many pupils in primary classes are rushed out for booster lessons. Some device to help them catch up also helps them feel valued and avoids falling further behind. I use catchup cards to help fill in code they have missed. They won't have as much understanding as those who have fully participated but their programs will work. This prevents them feeling stigmatised through never having anything finished.

This article originally appeared on Phil's blog (bit.ly/WhopUi). His website is www.code-it.co.uk/

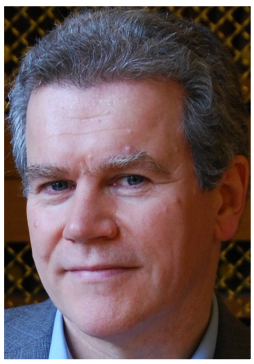
GOOGLE RISE AWARDS

Google's K12/Pre-Uni Education Outreach team announced that the 6th annual RISE Awards program launched in August. RISE Awards support organizations across the globe that promote extracurricular Computer Science education for girls, underrepresented minorities, and students facing socio-economic barriers. The portal will accept applications until September 30. Visit the website for eligibility criteria. We wish to partner with:

- organizations currently running extracurricular CS programs or
- STEM organizations looking to start a CS outreach program

Those selected receive grants between $15,000-50,000 USD and consulting services from various teams at Google for one year, with a focus on scaling their impact. *Marielena Ivory*

THE TURTLE SYSTEM: AN EASY WAY INTO TEXT-BASED PROGRAMMING AND COMPUTER SCIENCE



***The Turtle System*, with teaching resources and tools for setting and marking coursework, is available free thanks to a new project at Oxford University co-funded by the Department for Education. Peter Millican, Professor of Philosophy at Hertford College, explains the principles behind the system he has developed.**

PROBLEMS CROSSING THE POST-SCRATCH GAP

Visual drag-and-drop programming systems – notably MIT's *Scratch* – have proved extremely popular at primary level. Prior programming with *Scratch* should make text-based programming easier, by consolidating some essential concepts (e.g. variables and loops). But experience suggests that crossing the gap to textual coding is still a major challenge for both teacher and pupils, involving at least four difficult problems:

- Choice of language.
- Complications setting up and starting to program.
- Coping with syntax errors.
- Difficulty understanding non-visual program effects.

Programming language debates have gone on for decades: should you start with a language designed for teaching (like *BASIC* or *Pascal*), or with a commercially popular language (like *Java* or *Python*)? Industrial-strength systems can force novices to confront complexities of language far too soon, like the standard entry point for any Java program (which could almost be *designed* to intimidate beginning teachers and pupils):

public static void main(String[ ] args){

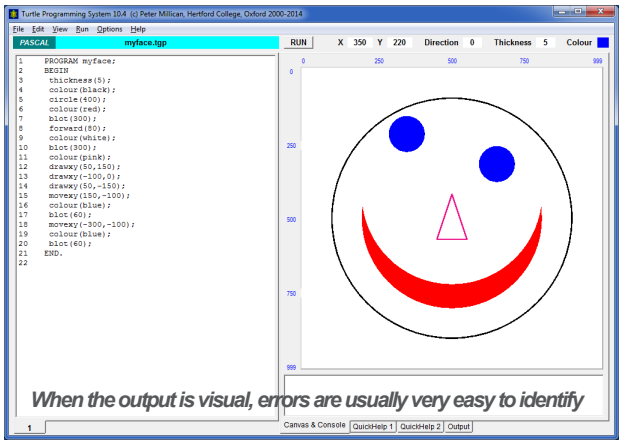
Moreover, such systems are set up more for textual than graphical output, so structures like loops and conditionals standardly get introduced using numerical examples. But for most people, these are far less easy to understand than pictures, especially when errors occur (see image right).

Text-based programming is notoriously difficult, both to teach and learn. But when I was teaching programming to novices at Leeds University in the 1990s, I found that by far the biggest problem was *syntax errors*. These were hugely time-consuming, as helpers rushed from student to student trying to identify the (mostly trivial) mistakes. Practical sessions often seemed like an obstacle course, "success" being defined more by stubborn endurance in the face of irritating and confusing obstructions than by any delight of creative achievement. No doubt this will be a familiar scenario for many readers!

The Turtle System aims to solve this and the other three big problems listed at the left. The last of them – deriving from our natural preference for visual effects – has a well-known solution: Seymour Papert's wonderful idea of *Turtle Graphics*. Papert's original version used his own language *Logo*, specially designed for the purpose, but add-on *Turtle* units have been created for many other languages, and are very widely used in teaching.

This suggests the idea of a simple, integrated environment designed to make Turtle Graphics as easy as possible, *using a general programming language of the user's choice*. Keeping things simple requires a "barebones" version of the relevant language, combining standard "core" features (e.g. constants, variables, arrays, subroutines, conditional and looping structures, operators and bracketing) with special features designed to make graphics and interaction straightforward. The core features allow standard programming techniques to be taught perfectly well. Built-in commands for graphics (e.g. **circle**, **colour**, **forward**, **print**), canvas control (e.g. **fill**, **pause**, **pixcol**, **update**), and simple access to keyboard and mouse events, make it very easy for beginners to get started and *have fun!*

Simplifying the language brings another great benefit, by *enabling error messages to be more precisely targeted*, because restricting users to a small number of core structures makes their mistakes easier to identify and correct. The current version of *Turtle Pascal*, for example, has around 150 syntax error messages, designed to point out *exactly* where the problem lies, and giving a precise hint for correction. After introducing my earlier version at Leeds, I found students were able to fix the vast majority of their syntax errors without needing any further help at all.



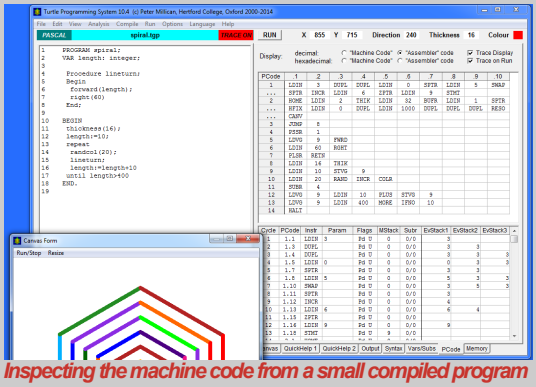
When the output is visual, errors are usually very easy to identify

The barebones approach also makes it feasible to offer a choice of languages (e.g. *BASIC*, *Java*, *Pascal* and *Python*), enabling pupils to compare the same algorithm in different languages, and ap-

LINKS TO COMPUTER SCIENCE: THE TURTLE MACHINE

In my first *Turtle* system (early 2000), programs were *interpreted*, i.e. read and executed line-by-line, but it proved very difficult to give well-targeted error messages within "nested" program structures (e.g. an "if" inside a "for" loop inside a "repeat" loop). I therefore wrote a *compiler*, so that the user's program would be translated into a form of *machine code* (or "PCode", short for "portable code"), whose instructions are executed when the program runs. Compiling involves a complete syntax analysis, helping to solve the error message problem. This also made it easy to provide information for teachers (e.g. about the structures and commands used), so we could check students' programs against specified requirements and mark coursework very quickly – a feature that will probably be appreciated by schoolteachers!

Compilation also opened a new possibility, of using the system to teach *Computer Science* concepts in a novel way. The compiled PCode – essentially a sequence of numbers – runs on a virtual (i.e. software-simulated) "Turtle Machine". These codes include simple instructions for moving, turning, drawing circles etc., and others for internal logic and memory operations. The latter (e.g. handling variables and subroutines) are quite sophisticated, providing plenty of potential for extension work (including finding ways to "hack" the Turtle Machine without any risk). But the basic drawing operations are very straightforward, and can be made open to view and even "traced" as they run, so *all pupils can learn about machine code by seeing how their own programs are analysed and executed in real time and with real effects.*



Inspecting the machine code from a small compiled program

preciate how all are translated into the same underlying "Turtle Machine" code (see box above). This code too is designed to be simple and understandable, making a virtual Turtle Machine easily portable to different devices. Running their own apps and games on the web and smartphones is exciting for pupils, and helps reinforce the vital lesson that algorithms can be understood quite independently of specific languages or hardware.

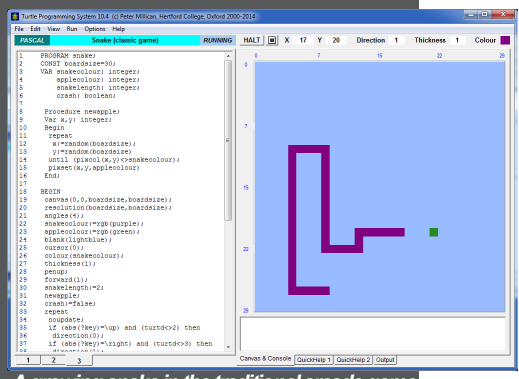
The language choice problem raised by the "Post-Scratch Gap" thus turns out to have a natural solution: teach pupils *explicitly* about the variety of computer languages, using barebones versions that make the comparison easy to understand. Pupils are greatly empowered by realising that programming skills are so easily transferable, and moving forward is far less intimidating when *problem style*, *programming language*, and *working environment* are changed one by one.

What pupils most need to learn is computational thinking and solving problems using algorithms. How those algorithms may then be expressed in some particular language is a secondary matter, because most computer languages are fundamentally very similar (far more than natural languages like English, French, German etc). Once pupils have learned how to express a simple algorithm within one syntax (e.g. *BASIC* or *Pascal*), it should be fairly easy for them to pick up another (e.g. *Java* or *Python*) within a day or two; moreover encountering this variety is, in itself, a valuable learning experience.

In the next issue of **SWITCHEDON**, I shall illustrate *The Turtle System*'s use in practical teaching, but in the meantime, it is freely available with plenty of teaching resources for introducing the new National Curriculum from www.turtle.ox.ac.uk. Please do take a look.

EDUCATIONAL BENEFITS OF BAREBONES SIMPLICITY

A huge amount can be done with a barebones language that has the core features listed, despite its relative simplicity. Illustrative programs provided with *The Turtle System* include a traditional "Snake" arcade game (65 lines), a "Paint" application (102 lines), implementations of famous cellular automata including the "Game of Life" (49 lines), and an infallible noughts-and-crosses program that uses the AI technique of recursive "minimax" analysis down the search tree (116 lines). Though a simple system, it can thus be used to explore algorithms that Computer Science undergraduates would study in their upper years at university.



A growing snake in the traditional arcade game

Learning on a barebones language is also entirely compatible with moving on to an industrial-strength version in due course (e.g. when starting A-level, university, or employment). Programming in *Turtle Pascal* or *Turtle Python* might not be quite the same as programming in *Delphi Pascal* or *Python*, but the basic algorithmic syntax and logic remain identical.

It is much easier getting into a full-strength system after the *Turtle* experience, crossing one hurdle at a time rather than having to learn about *both* computational thinking *and* a highly complex environment at the same time. If you have any questions, please contact me at peter.millican@hertford.ox.ac.uk.

SO STEPHEN SUTTON TAUGHT ME

Few people need an introduction to Stephen; his name, his fight against cancer, his motivational speaking and fund raising have made international news. In everything he has done, he has also shown us the power of IT to communicate and engage people for the good.

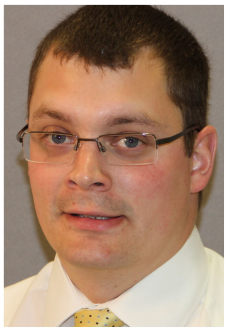
Stephen used Facebook, Twitter and YouTube really effectively to fund raise and engage people in his story. He created an e-book and a film of his short life. On April 21st Stephen posted a message on Facebook, saying it was " a final thumbs up". With his fundraising at £500,000+ and short of his £1,000,000 target Twitter and Facebook exploded. With everyone retweeting, celebrities soon got involved, from Stan Collymore, Clare Balding, Jason Manford and even David Cameron. Everyone willed the target to reach £1,000,000 before he died, and when his family tweeted that he had seen it, the world celebrated and extended the target to £2,000,000. It wasn't a final thumbs up, Stephen left hospital and lived a further 34 days. In that time an internet troll posted abusive remarks. Stephen took this in his stride, saying he was unaffected by the comments. The rest of the internet community rallied round and the account ended up being deleted. Stephen died on May 14, a poignant message from his Mum summed up everything and campaigns kicked in for various awards for Stephen, to a vigil at Lichfield Cathedral. Find out more, and donate at www.stephensstory.co.uk

TEENAGE CANCER TRUST

Stephen lived in Burntwood, and attended Chase Terrace Technology College; the school where I teach ICT/Computing. He used IT to connect with the world and did it quite brilliantly. He connected with people, everyone knew him - everyone was just in awe at his goodness. There is much to learn from this. We can teach students in school the power of the internet and social networks to do good and counter the often negative message given in numerous e-safety videos and lesson plans. Stephen has taught me that there is a positive message to be given to students and we can do more to harness their "connectivity" for teaching.

Louise Branch

A SIMPLE PROJECT UTILISING THE PYTHON TURTLE LIBRARY

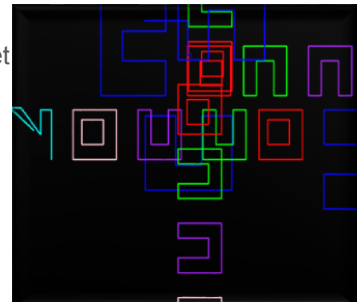


James Dent, Hertfordshire Hub Leader and Head of IT/Computer Science at Richard Hale School in Hertford introduces a project using Python turtles.

Never under estimate the power of the Turtle function in Python. During the summer term I was amazed by how much year 9 boys enjoyed developing screensavers of their names. Students started by building up their knowledge of the turtle by creating basic shapes. Start by giving them the basic commands such as `import turtle` and operations like `turtle.left(90), turtle.right(90),turtle.forward(50)`. Get the students to sketch their names on A4 squared paper, landscape using large block capitals and arc shapes, annotating sizes and then convert them to pixels. 1cm is approximately 50 pixels on screen. They can then calculate angles the turtle needs to follow. This will clearly get students analysing the problem, develop their numeracy, developing a solution and enable them to code more efficiently.

You need to make sure that the students are aware that the turtle always begins in the middle of the page and faces right. Make sure students do not call their file turtle.py as well. You may want to use code right to move the turtle to the top left. It took the students a bit of time to get used to moving between characters in their names using the `turtle.penup()` and `turtle.pendown()` functions. Lots of reference was made to the whiteboard and board markers to overcome this. Some students decided they wanted to change the turtle's drawing colour at different points. This started the conversation about HTML colour charts. The command students use to change the pen colour is `turtle.pencolor("#C38B8B")`. To turn name drawings into screen savers, students turned their code into a procedure and called it from the main code. They used the random number generator to change the start position for drawing names and a counter controlled loop (below). To get started a helpsheet is available on the CAS community (resources/2188)

```
import turtle
turtle.penup()
turtle.goto(-200,200)
turtle.pendown()
```



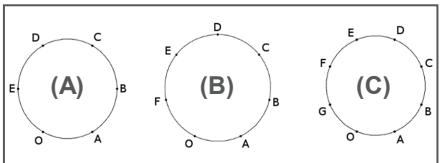
```
import random #import the random Python library
turtle.goto(random.randint(1,300 ),random.randint(1,300))
#Select a random start position
for start in range(2000):
    turtle.bgcolor("black")
    turtle.speed(10)
    turtle.penup()
    turtle.pensize(3)
    turtle.goto(random.randint(1,300 ),random.randint(1,300))
    turtle.pendown()
    myname()#This calls the procedure the student created
```

REACHING FOR THE STARS TO HELP DEVELOP COMPUTATIONAL THINKING

Programming is a very effective way to open up symmetry and pattern, which underpins much work in turtle graphics. CAS Master Teacher, Dave White explores ways to develop computational thinking by developing 'unplugged' investigations in Python.

Whether Logo, Scratch or Python, many teachers see the immediacy of the graphics screen as a creative way to introduce the fundamentals of programming. Drawing geometric shapes like squares, polygons, circles, stars, spirals and on to generating unique patterns creates an aesthetic buzz. Recently, I became interested in scaffolding the learning of programming for beginners with 'unplugged' activities which weren't just about the relevant geometry of the shapes, but which included activities which mimicked the motion of the turtle/sprite: *action geometry*, I call it.

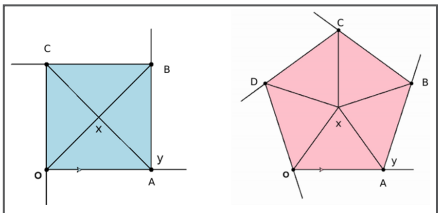
I liked the mystery of joining dots as a kid. Let's see, 4 dots in a square, 5 dots: a pentagon or pentagram (star). How do you draw a pentagram from five dots, and can you do it without taking your pen off the paper? Similarly, drawing the 5 dot envelope. Are these my secret childhood *algorithms* before I knew what the word meant? 6 dots: a hexagon or hexagram (star) - you can't draw this one without taking your pen off the paper. 7 dots: heptagon and different heptagram stars… How about putting the dots on a circle? How about illustrating the structure of a polygon/star in diagrams. How about inviting the pupils to draw shapes on paper in the same way that the turtle draws them on the screen? I was on a roll…



Drawing Polygons: (A), (B) & (C)
Start at O, join dots OA, AB, …

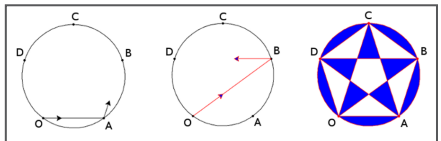
Drawing Stars:
(B) Start at O, skip a dot, join OB, BD …
(C) Start at O, skip 2 dots, join OC, CF …
(A) Start at O, skip a dot, join OB, BD …
What happens?

I got out my drawing tools: pencil, paper, ruler, compass, protractor, set square, to make up the templates; and I struggled. Have you ever tried drawing 4 equally spaced dots on a circle, let alone 7? I felt like Archimedes must have felt when trying to square the circle in his quest for π (pi). Then I realized I had what Archimedes didn't have: a programmable computer. This felt like cheating, but I got over it. I could produce the unplugged learning materials for learning to program by programming. A variation on the chicken and the egg, but it wasn't easy - cheating never is!



Action geometry for a square and pentagon

Writing a program to draw the simple dot shapes needs little more than the skills of a beginner who has learned how to draw polygons; but more computational thinking and some basic trigonometry is required to draw the combined shapes that I use in my 'unplugged' sessions for teaching programming. If you want to have a go look at the diagram above - the trick is to do the square first to get an idea of what it's about. Find its centre... then generalise to the pentagon.



Action geometry for a pentagon and pentagram

A further tip, but only for the pentagon: the ratio of the length of the pentagram(star) side to the length of the pentagon side OB/OA is Φ (phi), a magic number in art and design.

The teaching aids (Python 3) can be downloaded from my website. Producing 'unplugged' learning aids is worthwhile project material for pupils (and teachers!) at different levels from KS2 up to GCSE. Suggestions, hints and solutions (including a search for π with Archimedes, and Φ with Phidias and Fibonacci) can be found on ispython.com. Thanks to Archimedes, and Mark Dorling for his work with Scratch and Logo for the inspiration.

CS4FN FEATURES MAP TO COMPUTATIONAL THINKING

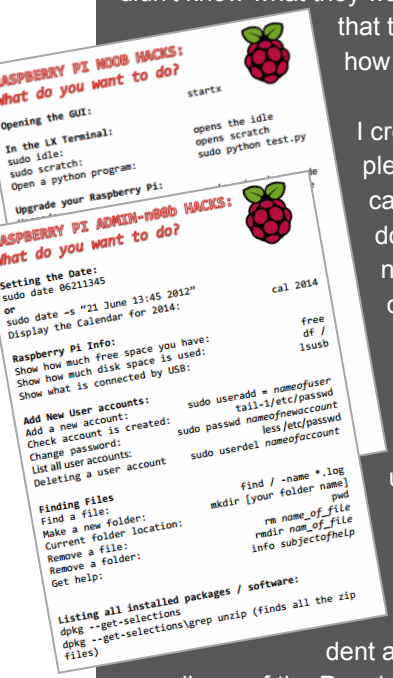
Teaching London Computing, cs4fn's sister site of fun teacher resources, based at Queen Mary University of London, has new resources to support teachers. We've organised our cs4fn articles and our fun unplugged activities around computational thinking themes: Algorithmic Thinking, Evaluation, Decomposition, Abstraction and Generalisation. Get a deeper understanding of their meaning and links to leading edge research by reading cs4fn!

We've also created a cs4fn version of the CAS progression pathways. An interactive pdf maps each pathway to cs4fn articles. Click on a particular level, in a pathway such as Programming & Development or Data & Data Representation, and you will be taken to a collection of fun articles related to it. Visit teachinglondoncomputing.org to find out more. *Paul Curzon*


Connecting Scratch to outputs via the Cumbria breakout box

GET STARTED WITH LINUX AND RASPBERRY PI CHEAT SHEETS

Dan Aldred, Curriculum Leader at Thirsk School & 6th Form College found that beginning to teach and use the Raspberry Pi was an exciting journey that many students were ready to embrace and were inspired by. A passion for stimulating learning and developing others is important for fostering intrigue and progress but the biggest stumbling block students came across was remembering all the shell commands. Much of the lesson would be spent trying to figure out the correct line or remembering missing out a syntax. The issue was never that students didn't know what they wanted to do, but that they didn't know how to do it.



I created these simple A5 reference cards. They are double sided, laminated and given out with the Raspberry Pi. It has enabled students to quickly look up commands and use the Pi and they now recall many of the commands. They are now more confident and faster in their overall use of the Raspberry Pi.

As they furthered their understanding I introduced the admin cheat sheet. This contains advanced shell commands and again acts as a quick reference tool. The You can find cheatsheets on the CAS Community at: resources/2132 and resources/1193 *Dan Aldred*

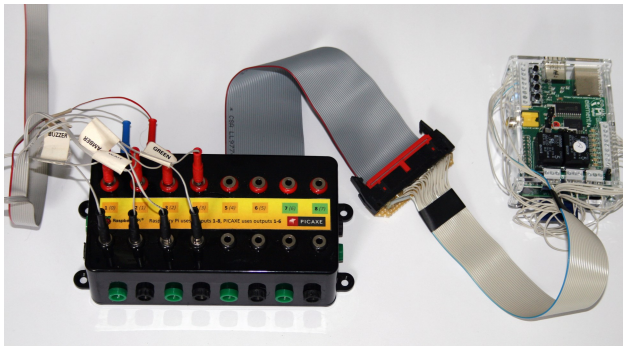
DEVELOPING SUPPORT FOR THE PITO TEACH COMPUTER CONTROL

A visit to a Manchester CAS event made Mike Wilkinson, faculty technical manager at the University of Cumbria, think about how the Raspberry Pi can be effectively demonstrated in schools with real world applications.

After seeing an inspiring demonstration of a Lego crocodile using Scratch at a Manchester CAS event my technical team began to look at the practical aspects of using the Pi to control everyday items such as lights, motors and sensors. We purchased a Piface which connects to the Pi allowing easy addition of external devices. Scratch was an obvious choice of programming tool. Using the two together on a Pi we quickly identified two main problems. First, the Piface/Scratch combination slowed things down so much it became unusable. We contacted the team behind the Piface, Dr Andrew Robinson and colleagues at the University of Manchester (www.piface.org.uk/about/). After some much appreciated work on the interface they solved the speed problem.

Secondly, we had a number of physical challenges with the hardware:

- Piface screw connectors avoid the need to solder things. However, connecting a model with numerous inputs and outputs resulted in a complex jumble of cables. Colour coded cable helped but it was still time consuming and error prone to connect several items. Whilst perhaps desirable to teach older/more able children fault finding, we felt struggling with the basic setup could disengage students at an early stage.
- Getting things to work together can be technically challenging! We wanted to show a cross-curricular application using a simple powered model car that you might see as a school D&T project. However, simple DC motors do not connect directly to the Piface, needing an extra motor control chip in the circuit.
- Connecting old control equipment. Like many schools, we have a number of old, yet relevant, models which previously connected to old control boxes. These could be repurposed.



To address the problems we created a breakout box, allowing very easy connection of several devices to the Pi. It uses standard connectors and sockets, numbered to match the Piface outputs/inputs and scratch program control IDs. Additional circuitry needed is hidden in the 'black box'. The Pi and Piface cables are wired to a permanent socket which the connects to the box cable. Model cables connect to the box sockets using standard plugs. Everything goes together in a fast, simple and error-free way. We hope this fast assembly will allow better focus on the application of the technology rather than a struggle with technical issues. We intend collaborating with school teachers to see if this approach really does encourage increased engagement with real world application in computer science. It's a small step but the future possibilities are huge!

A MODEL OF PROGRESSION WHEN LEARNING HOW TO PROGRAM

At KS3 pupils should be exposed to at least two programming languages, one of which should be text based. Tim Eaglestone, a computing teacher at Dorothy Stringer High School, Brighton argues that we should treat the early stages of coding as a literacy exercise.

I've been working on an approach to teaching programming that draws on standard methods of getting pupils to engage with written texts. In doing this, I've found a need for an underlying model of progression - or taxonomy for want of a better word - to help me think about their learning. The idea is that the majority of pupils will only just be beginning to write their own fragments of code by year 9. Hopefully along the way, some will be switched on to programming, but all pupils will have experienced working with code to a level that suits their ability. The taxonomy currently looks like this:

Spotting keywords:	A good editor with code highlighting helps as you can start with the question: "What's different about the way these words are written?" and follow with why questions or an explanation. Pupils can progressively build a dictionary of keywords for your language, or use things like 'Ever-heard-the-word' grids.
Identifying numbers and data (variables)	Ask what a number in a particular line does. Some will be variable assignments, some might be numbers that are hard coded for some reason.
Spotting syntax elements (such as block starts and ends)	Looking at how code is structured, I find I need to be very explicit about where blocks of code start and end. I often need to spell out the role of parentheses and different brackets. There is also a good opportunity to discuss techniques such as indenting, comments and good variable names to make code easier to read for humans.
Recognising constructs	Now they can spot elements, can they identify different types, insert or read sequences, loops or conditions?
Tracing (running through the code on paper step by step)	Read aloud, line by line, or write in a book what each line does of a given piece of code. (Line numbering on your editor/ print-outs is important to help this.)
Explaining the code in English in the form of a trace	"First it does this, then it does that…" A simple trace explains each line or command, but do they understand the algorithm? A good question to ask is how it could be tested?
Explaining the code in English in terms of its function	E.g. "This code sorts an array from smallest to largest". A harder activity that requires a general understanding of the code. It is also a good opportunity to use technical language, and allows questions about testing strategies.
Using diagrammatic abstractions	I've found drawing a flow-chart first then coding to be confusing for some. It seems like there is a stage involving matching code to flow-charts to reinforce an understanding of the algorithm for program flow.
Conservation exercises (e.g. "This method is not finished. Complete it so that it …")	Offer an algorithm or expected outcome, provide some code fragments or stubs for them to complete. Or pepper some code with alternatives for the next line and get them to choose. Debugging exercises can also help this.
Reversing exercises	"Here's some code that does X, Y and Z. Write some code that does the reverse: -Z,-Y then -X." Coming up with these can be tricky but they do test pupils' understanding and encourage them to produce code of their own.
Inference exercises involving holding several possible program states in mind to answer a question.	A pupil may be asked to generalise a system's response by reading a fragment of code Holding several states or a more complex model of program flow in mind is a good test of understanding and aptitude for programming. It's also an important stage in their understanding of abstraction with regard to developing code. This step is also where the pupils are starting to code more independently .

The first four steps aim to help pupils gain confidence when presented with code and provide strategies for exploring it. The tracing through to diagramming steps are essentially about comprehension of the individual elements that leads onto thinking about algorithms and program flow. It's only in the final stages where they start to produce their own lines of code.

It's worth pointing out that I don't see this as the progression for the Key Stage. Rather, it is iterative. For example, teaching students to draw using turtle graphics procedurally is one thing - many will advance to the higher ends of the taxonomy quite quickly given a small set of commands and limited exercises. Introducing the idea of functions or subroutines would involve them learning the syntax from the earlier stages of the taxonomy and working up as they apply it.

The rate at which they progress through the taxonomy seems to be in line with their confidence and experience. I have also found this model to be useful when thinking about where pupils get stuck. A big barrier for some is the apparent pedantry of the programming language syntax and the need to accurately enter code.

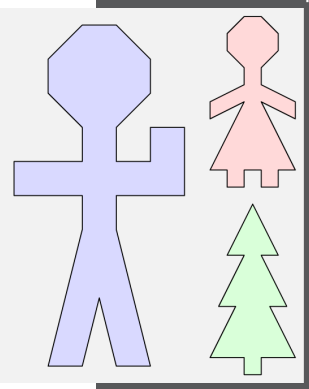
Giving exercises from the beginning of the taxonomy can help, but it is important that they also get the chance to experience getting code working for themselves. Examples of exercises used in class can be found on my blog, in a presentation given at the Brighton and Hove Joint Practice Day. It is a work in progress and I hope it will be of some use to others. I would welcome any comments or observations to help me develop it further.

This article first appeared on Tim Eaglestone's blog, and can be found at www.timeaglestone.co.uk/category/teaching/

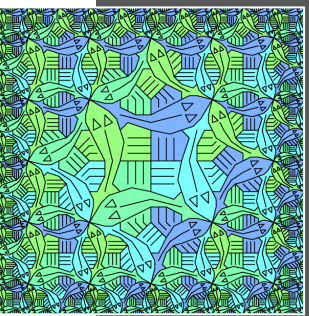
PICTURING A DIFFERENT PROGRAMMING STYLE

There's a style of programming, and programming languages to support it, where many of the basic constructs we take for granted in procedural programming aren't used. The opportunity to view programs as more than a sequence of commands makes functional programming a valuable tool to encourage students to think about programming.

Mike Spivey, from the University of Oxford developed GeomLab,



initially used for the Gifted and Talented programme to give young students a taste of functional programming. An online activity, it combines functional programming with graphics. The basic functions of the GeomLab language can be used to assemble pictures from pre-defined tiles. The expression `man $ (woman & tree)` displays a man placed beside ($) a woman above (&) a tree.



A set of progressive exercises challenge pupils to describe more complex pictures using recursion. Eventually they find themselves investigating Escher like images which can be created from a few basic tiles. GeomLab (www.cs.ox.ac.uk/geomlab) provides an accessible introduction to the ideas of functional programming, be it for a lesson or an extended investigation. There are other approaches to programming too, and the wider the variety children encounter, the richer their experience will be. *Roger Davies*

FUNCTIONAL PROGRAMMING: A DIFFERENT PARADIGM

There are different approaches to writing programs. With functional programming now featuring in some new A level specifications, John Stout offers some timely examples.

The examples in Figure 1, top right, show how to square every element in an array (VB.NET) or list (Python), and put only the odd elements of `ints` into another array. When we use *functional programming* we can apply a function (`square`) to the array or filter the elements of the array using another function (`odd`):

```
Dim result = ints.Select(square)
Dim result = ints.Where(odd)
```

In VB.NET, `Select/Where` are methods that can be called on any enumerable (containing a number of things that can be selected one by one in order) object.

```
result = map(square, ints)
result = filter(odd, ints)
```

Similarly Python uses the `map/filter` functions which take a function and an iterable. However, often you don't have the function that you want to give to `Select/Where` or `map/filter`, and having to define it just for the one job is awkward. Enter *anonymous functions* (or *lambda*s).

In VB, rather than using a predefined function, you can define the function you 're using as you use it (see Fig 2 below). You don't need to define the type of `i` since it's only ever going to be passed an integer (each one from `ints` in turn). Python allows the same thing, but uses the technical term `lambda` (from the lambda calculus).

Since functional programming treats functions as *first class values*, just like integers or strings, if you use a function more than once you can define a variable to hold the function. This is the way `square` has to be defined in VB to work with `Select/Where`, or Python for `map/filter`.

```
Dim square=Function(i As Integer) i*i
square=lambda i: i*i
```

That `result` is not (at least not immediately) an array or list turns out to be a positive advantage. Instead `result` is something that will, if you 'ask it nicely', give you back the items of the result the rest of the program needs. Getting the results is delayed until you need them. This is called *lazy evaluation* in that only as much work as needed is done. Primes in Haskell (far right) is a beautiful example.

It may be that you don't need all of the results, just the first one that satisfies another condition. Study the example above right in Fig 3, which gets the first integer from `ints` whose square is greater than 8. This executes the squaring function 3 times (for i = 1, 2, 3) because as soon as 3 gets squared the result is 9 (greater than 8). The (0) at the end means that only the first number in the result is needed. Using (1) would need two elements to be returned, so 4 squared would be evaluated as well. It may not appear to be saving much but for more complex functions or larger arrays it could be.

Simple ways to force evaluation include printing them one by one, or just asking how long the result is (using `.Count()` or `len()`).

```
Dim result = ints.Select(Function(i) i * i)
Dim result = ints.Where(Function(i) i Mod 2 = 1)

result = map(lambda i: i * i, ints)
result = filter(lambda i: i % 2 == 1, ints)
```

Fig 2

Figure 1

VB.NET

```
Dim ints() = {1, 2, 3, 4, 5}
For i = 0 To ints.Length - 1
    ints(i) = ints(i) * ints(i)

Next

Dim oddInts(ints.Length) As
Integer
Dim j = 0
For i = 0 To ints.Length - 1
    If odd(i) Then

        oddInts(j) = ints(i)
        j = j + 1
    End If
Next
```

Python

```
ints = [1, 2, 3, 4, 5]
for i in range(len(ints)):
    ints[i] = ints[i] *
ints[i]


oddInts = []

for i in range(len(ints)):
    if odd(i):

oddInts.append(ints[i])
```

Fig 3

```
n = ints.Select(Function(i) i * i).Where(Function(i) i > 8)(0)
n = list(filter(lambda i: i > 8, map(lambda i: i * i, ints)))[0]
```

FUNCTIONAL PROGRAMMING: AN EXAMPLE IN VB

This is given in VB.NET only, for reasons of space. For parents' evening every Teacher has a property (apps) holding a list of Appointment slots in time order, each of which is 5 minutes long, and whose student property holds Nothing if the appointment is free, or the Student being seen if not. Teachers can make appointments for multiples of 5 minutes with certain students. Ts and Ss represent the list of teachers and students, T and S one teacher and student.

Does teacher T have a run of at least n free slots?

```
T.apps .SkipWhile(Function(app) app.student IsNot Nothing)
       .TakeWhile(Function(app) app.student Is Nothing)
       .Count() >= n
```

Can teacher T be seen by a parent immediately after sTime for n slots?

```
T.apps .SkipWhile(Function(app) app.start < sTime)
       .TakeWhile(Function(app) app.student Is Nothing)
       .Count() >= n
```

How long is student S being seen for by teacher T (assuming a student's slots are contiguous)?

```
T.apps .SkipWhile(Function(app) app.student IsNot S)
       .TakeWhile(Function(app) app.student Is S)
       .Count() * 5
```

What fraction of teacher T's appointments are taken?

```
T.apps .Where(Function(app) app.student IsNot Nothing)
       .Count() /
T.apps .Count()
```

Which students is teacher T seeing?

```
T.apps .Where(Function(app) app.student IsNot Nothing)
       .Select(Function(app) app.student)
       .Distinct()
```

And so on

`SkipWhile`	skips elements from the start as long as the function returns True
`TakeWhile`	only takes elements from the start as long as the function returns True
`Count()`	return the number of elements and
`Distinct()`	only return distinct elements.

Both force the complete evaluation (otherwise they can't produce an answer).

PRIMES IN HASKELL

In a true functional language such as Haskell, there are normally no side-effects of evaluating a function, and with the same input parameters a function will always return the same value.

The *Sieve of Eratosthenes* (bit.ly/1xB5peK) is an algorithm for printing all the prime numbers up to a value n:

- Start with a list containing the numbers from 2 to n
- Print the first number (call it p) in the list (it's a prime)
- Remove from the list any number that's a multiple of the first number, i.e., p, p+p, p+p+p, p+p+p+p, etc
- If there are any more numbers left in the list repeat from step 2
- Stop; you've printed all the primes less than or equal to n.

A beautiful Haskell implementation of this algorithm is

```
eratosthenes (prime:rest) =
prime : eratosthenes (rest `minus`
[prime, prime+prime .. ])
```

When executed with the command `eratosthenes [2 ..]`

- `(prime:rest)` splits `[2 ..]` into its first element, `prime (2)` , and uses `rest` to refer to the rest of the list `([3 ..])`
- `prime :` on the righthand side of the = sign constructs a new list with `prime` as the first item and what you get back asking `eratosthenes` for the primes in the list `rest ([3 ..]) `minus'` the numbers in the list consisting of `prime`, `prime+prime`, `prime+prime+prime`, and so on.

Due to *lazy evaluation* the elements in the infinite lists `[2 ..]` and `[prime, prime+prime ..]` aren't created until the program needs them, and then only as many as the program needs. When you start this running the act of asking for the value of `eratosthenes [2 ..]` forces the primes to be displayed, 1 by 1.

PROGRAMMING CHALLENGE SUCCESS

Programming Challenge 4 Girls, an international event originating in New Zealand, has spread to over 12 countries in just five years. The aim is to give girls (13-14 years) the opportunity to work together on an engaging project using Alice. PC4G is designed to be approachable, fun, challenging and educational. Year 9 girls at Fakenham Academy signed up and after a short introduction, went off to refresh their Alice skills. Two Alice worlds had been created for them. With clear storyboards and instructions on how to complete the animations the girls had two hours to complete as much as they could. To help them they had short films showing examples of completed animations.

It was truly wonderful to see the girls so fully immersed in the activities for the whole morning. Rarely does one find an activity so engaging and engrossing! The girls worked in pairs, helping each other and other teams; the sharing and collaboration was fantastic to see.

This year we were joined by Margot Phillipps the Director of PC4G who was visiting Europe on her mission to spread PC4G. We were also joined by Dr Pam Mayhew, Senior Lecturer in the School of Computing Sciences at University of East Anglia. In New Zealand PC4G usually runs at venues such as universities or higher education colleges. Teams of girls attend from local schools with regional winners going on to a code camp at Auckland. Pam was very impressed with the girls' work and level of challenge offered by PC4G – hopefully the School of Computing at UEA will be able to take this on in the future.

During lunch, with a strict set of criteria laid down and clear marking structure, staff assessed the work. The girls reconvened in the hall for the end of the day. Everyone received a Certificate and winning teams awarded 25 medals. At the end of a very busy and hectic day the girls went home very excited and keen to show off their medals and certificates! *Sue Gray*

UNDERGRADUATES CREATE 'CODING CURRICULUM' AT UCL

Professor Stephen Hailes and Dr Dean Mohamedally, from UCL's Department of Computer Science asked their undergraduates to work in teams and create resources to support teachers introducing the new curriculum.

They called the activity 'UCL Coding Curriculum'. Who better to spread an enthusiasm for coding and computer science than the undergraduates who are much closer in age to the students in schools? Overall, the hard work and creativity of the UCL teams really shines through in the outstanding work they have produced. In one project, Python is introduced using the theme of Cryptography and Ciphers; the material contains unplugged activities and exercises enabling students to familiarise themselves with the concepts involved before plunging in with coding. In another, the material covers building dynamic web pages using HTML5, CSS and JavaScript. All of the resources are standalone, in that students can work through them at their own pace. This makes them ideal extension activities for those who want to learn more.

The top seven teams were awarded certificates by Dalim Basu, Chairman of BCS North London Branch, supporters of the project, in an awards ceremony last June. The projects are published on the UCL Computer Science Outreach webpages and have also been added to the CAS Community (use UCL to search the resources). The benefits of the project are two-way. For undergraduates, learning how to teach is a valuable skill which can also provide an insight into the ways in which we accumulate knowledge. UCL are looking forward to repeating the activity next year, this time placing undergraduates in local Camden schools so that they can work directly with teachers as part of the Get Camden Coding campaign. So far the resources have received a very positive response from teachers who have commented on the high quality of the materials.

Current undergraduates Rohan Kopparapu, Delia Gander, Justin Sibiescu, Chaitanya Agrawal and Vardan Tandon commented on their experience: "Encryption is one of those things in computer science that goes unnoticed and is usually underappreciated. Yet, it is also one of the most interesting and important fields of study. We feel that teaching programming unconventionally while using ciphers as the main focus is an excellent idea. It would help strengthen the students' programming foundation without boring them to sleep, while giving them an insight into ancient ciphers as well as modern RSA and quantum cryptology. We hope this inspires a generation of students to take a greater interest in the fast growing industry of computer science." *Rae Harbird and Stephen Marchant*

NORTHERN IRELAND COMPUTING AT SCHOOL CONFERENCE SUCCESS

An exceptional day of computer programming, creating computer games and sharing best practice was how the inaugural conference for 'Computing at School in Northern Ireland', hosted by Stranmillis University College, was described by delegates.

Mark Dorling from CAS(UK) advised us to 'Hack The Curriculum!' and provided a high octane demonstration of 'Speed Lessons' providing fun, creativity and cross-curricular ideas for teaching computing. Conference delegate Chris Robinson commented that: "Mark gave the audience an insight into what has made him such a leader in his field, it was hard not to feel inspired and indeed challenged to go back to school with a new set of ideas on how we can integrate ICT into the curriculum. A great start!"

While attending the conference Mark was delighted to meet and congratulate Ruth Foster from The Wallace High School, Lisburn. Ruth is the first teacher in Northern Ireland to receive accreditation through the BCS Certificate in Computer Science Teaching.

Conference delegates were spoilt for choice in selecting their workshops from the varied list of possibilities. There was everything from how to make a banana piano with Makey Makey through to making games using Scratch and Construct 2 (courtesy of the 'NERVE Centre' Belfast) to advice on professional development.

It was inspirational to listen to the EdTech Winners of 2013; Gareth McAleese (GoBeserk), Roisin Crawford (STEM Aware) and Kerri McCusker (PhD Researcher, University of Ulster). These three young participants related to delegates findings from their recent US funded exchange program. There were some interesting tales on what is happening in American classrooms and fabulous illustrations of work we can do with 'jumping robots', made from polystyrene cups, plotting out their travels and using mystery Skype to connect classrooms around the world.

The final lecture was from the inspirational Stephen Howell from Microsoft (Ireland). For most delegates it was the first time that they had received a sermon from a 'Technological Evangelist'! We were encouraged to travel from Minecraft to Mind Crafting - new tools to help you teach Computational Thinking. It was easy to see why Stephen is internationally recognised for his work and presentational skills as he demonstrated to conference delegates Kinnect2Scratch. Developed by Stephen and available at scratch.saorog.com, it means anyone can write a program with motion control, use gestures, make kinetic games and have fun as they become active in their own game.

One delegate commented: 'I have to say that the conference left me feeling extremely excited at the real untapped potential that we have here. With regard to the children and young people that we can inspire on a daily basis, the sky is the limit. Literally!'

The conference, organised by Dr Irene Bell, the Chair of CAS (Northern Ireland) and Head of STEM at Stranmillis University College was supported by Kainos, CEM Systems, Siemens and Intel. *Irene Bell*

SCOTLAND EMBARKS ON

PLAN C

Professional Learning And Networking for Computing

PLAN C is a 2 year programme funded by the Scottish Government and overseen by the BCS Academy and CAS Scotland to support Computing teaching. One main strand focuses on developing new pedagogy and supporting activities for National 4, 5, and Higher Computing Science. Education research has shown some strategies help a wider range of students understand and apply the core ideas.

With the training of 32 PLAN C lead teachers now complete in local authorities across the South and Central Scotland many local hub groups are starting up to share experiences, activities and provide mutual support. Registering to participate in a local group has a number of benefits:

- learning specifically targeted at why beginners find some concepts more difficult. A focus on changing teaching so more experience success in the new Computing Science qualifications.
- an online space with background research, presentations, templates and example activities.
- a supportive group that meets face to face with a focus on sharing experiences to improve.
- a structured programme of sessions with accreditation at the end by the BCS Academy of Computing. This will also allow you to meet many of the new requirements for professional update.

Further details are available on www.planforcomputing.org.uk but crucially your participation in PLAN C is an opportunity to help reinvigorate and rejuvenate CS in Scotland for a new generation of learners. *Peter Donaldson*

CONNECTING THE CURRICULUM TO CAREERS IN TECHNOLOGY

Leading UK organisations have launched the CREATE! Campaign. The campaign provides educators with free programmes to help bring their STEM, computing, economics, enterprise and careers curriculum to life - through events with existing entrepreneurs, an 'app creation' competition and additional initiatives at the intersection of innovation and enterprise. Most jobs in the UK are created by companies less than five years old, with over three million new jobs projected to be created by the app economy alone in the next five years.

THE CREATE! CAMPAIGN

The campaign hopes to inspire students to create their own solutions to challenges they see around them, and to provide a clear link to how the skills they are learning in school can be used to create the businesses of the future. Sherry Coutu, Chairman of Founders4Schools, one of the contributing partners said, "We want to help teachers show students where the future jobs are, and encourage students to develop the skills to fill and create these jobs."

Teachers can work with Founders4Schools to search for and invite founders of fast-growing, successful businesses into their classrooms to talk to students about entrepreneurship. This programme can be linked to an immediate real world application by entering students into the CREATE! Your Own App Competition to develop their own idea for an app. Winning app teams will be invited to an awards ceremony in London during Global Entrepreneurship Week.

To continue the momentum, providing students with initiatives to connect technology and enterprise, teachers can find out about the work of organisations like Apps for Good, Behind the Screen, TeenTech and Young Rewired State. Additional partners for the campaign include Codecademy, Decoded, Code Club, iDEA, Raspberry Pi Foundation, Make Things Do Stuff, Cambridge University Press UK Schools and more. Teachers can get further details and sign up online at the CREATE! Website: www.createcampaign.org *Rajal Pitroda*

INFORMING CHOICES WITH CAREERS GUIDANCE

Since 2012, schools have responsibility for providing access to impartial careers guidance. Bruce Nightingale, who teaches in Staffordshire, explains some implications.

The advice schools offer must include information on all 16-18 education or training options, including Apprenticeships. Your school may have a careers adviser but, as the statutory guidance makes clear, schools need to supplement this with external sources of careers guidance to meet the new duty. This could include an external careers provider, employer visits and mentoring. Collectively, the sources must provide information on the full range of post-16 options and access to face-to-face support where needed.

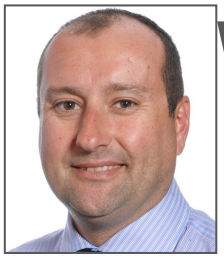
Ofsted published 'Careers guidance in schools not working well enough' in September 2013 which highlighted issues in the arrangements for careers guidance in schools. Three quarters of the schools visited for the survey were not implementing their duty to provide impartial careers advice effectively. The survey found that guidance for schools on careers advice was not explicit and there was a lack of employer engagement. It goes without saying that vocational education can facilitate routes into employment, but what is often poorly understood is the role that apprenticeships and traineeships offer young people.

Why might a young person want to study BTEC level 3 computing qualifications as opposed to GCE A levels? Level 3 (and Level 4) BTEC in Professional Competence for IT and Telecoms Professionals (QCF) qualifications include vendor Units such as Introduction to Cisco Networking Technologies, Interconnecting Cisco Networking Devices, Cisco Exploration Network Fundamentals, CompTIA Linux+, CompTIA Security+, CompTIA Network+, CompTIA A+ Essentials, CompTIA Server+ and, of course, the Microsoft Certified Professional units.

The content of Level 3 BTEC offer students wide unit choices such as Managing Networks, Software Design and Development, Organisational Systems Security, Computer Networks, IT Technical Support, IT Systems Troubleshooting and Repair, Computer Systems Architecture, Maintaining Computer Systems, Client Side Customisation of Web Pages, Web Server Scripting, Website Production, Installing and Upgrading Software, Developing Computer Games, Event Driven Programming, Object Oriented Programming and Procedural Programming

The practical, applied nature of vendor qualifications enjoy global recognition amongst employers, whilst BTECs are recognised and valued by UK employers. Where do Apprenticeships fit? If your pupils want work that involves 'doing' any of the above, then apprenticeships might get them started. To see current (real time) opportunities, go to www.apprenticeships.org.uk/. Search 'vacancies and opportunities' - try keyword 'computing' and select your location e.g.'computing' and 'Greater Manchester' found 700 possible apprenticeships. In the next issue we shall look at the role of training companies and employers in the provision of apprenticeships and traineeships and meet several young people as they embark on an IT/computing apprenticeship.

COMPUTING CYBER APPRENTICES VISIT JAMK IN FINLAND



Malvern is rapidly becoming the centre of several companies specializing in fighting internet crime. John Palmer, from The Chase, Malvern outlines an innovative scheme to get students involved.

Malvern, at the centre of the CAS '3 Counties' (Worcestershire, Herefordshire and Gloucestershire) Hub has been a centre of Computer Science expertise for decades. It is a place where innovation thrives. Baroness Neville-Jones, ex-Chair of the Joint Intelligence Committee/QinetiQ and a champion of cyber security, believes there will be hundreds of thousands of new jobs in this area created in the UK over the next 10 years. The Cyber Security Apprentice Development Scheme (CADS) is an innovative programme between established local Cyber companies QinetiQ and 3SDL, and The Chase, Hanley Castle and Pershore High Schools in Worcestershire. These high schools are very enthusiastic and active members of CAS '3 Counties' with excellent reputations for producing high quality students of Computing and other STEM subjects. CADS meets up for one afternoon every week at the National Cyber Skills Centre on the Malvern Hills Science Park.

The aim is to encourage sixth-formers with a talent for Computing and an interest in 'ethical' Cyber Security to come straight into the workplace at age 18 after their CADS internship and earn whilst they learn (modern apprenticeship). This 'apprentice' programme has nothing to do with tool bags, making tea and sharpening chisels, but is for candidates whose career path will be as stellar as a graduate's. In return no student debt, high quality work, a good salary and travel, achieving their degree at around 25 rather than 21.

CADS has forged links with Jyvaskyla University of Applied Science (JAMK) in Finland, and we are now collaborating with them on a number of interesting cyber projects. Finland has centred its cyber skills development around a single facility at JAMK. Whilst on the fringes of Europe geographically Finland is central to the EU cyber security agenda.

In April, members of CADS were invited to Jyväskyla to visit JAMK where we learnt about the work of the University in the area of Cyber Security and also about life in Finland. We talked to some of the students about the projects that they were working on; it was interesting to find that many of these were based on Open-Source technology such as Python, Linux and Raspberry Pi. We also viewed their security exercise room, which is only available to the students on the Cyber Security courses; we were shown a virtual fully simulated internet – modelling ISPs of each nation/continent across the whole globe. The importance of this modelling means that JAMK can simulate geo-located attacks and measure the estimated load on network cables and stress on the ISPs. We were also shown a live simulation of a DDOS attack on a hosted site in the virtual network. This helped us see how a security company might model and plan for ways to fend off malicious cyber-attacks against either corporations or ISPs. Although the trip was mainly focussed on Cyber Security time was taken out to experience Finnish culture with a JAMK student currently working in the UK at 3SDL!

David Willetts, the Government's then minister for Universities and Science, visited Malvern last term (see pic, bottom left) to launch the opening of the new National Cyber Skills Training Centre. He was accompanied by West Worcestershire MP Harriett Baldwin. A number of students from The Chase were there to welcome them and talk about CADS. Holly Salt from Year 11 also explained how The Chase is leading the way in developing the new GCSEs in Computer Science. The National Cyber Skills Centre aims to close the skills gap in this rapidly growing industry. We hope that many more sixth-formers in the CAS '3 Counties' area will benefit from their enthusiasm and expertise of employers such as 3SDL and QinetiQ in the future. There are many different, interesting and exciting routes into rewarding Computing careers.

gocracker **CAREERS ADVICE FOR STEM SUBJECTS**

Gocracker is designed to encourage young people to study STEM subjects and to discover the exciting career opportunities available to them in science, computing, engineering and other technology industries. Young people can find information and news about STEM subjects, industry sectors, apprenticeships, universities and leading employers. See www.gocracker.com

A PAUSE FOR THOUGHT

20 Questions began as parlour game in the 19th century, developed as a radio programme in the 1940s, then TV, and spawned many variants eg the Guess Who board game. 20Q, the website (www.20q.net/) based on the game, originated as an artificial intelligence (AI) test in 1988.

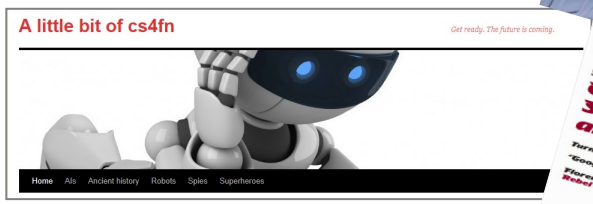
A useful strategy is to ask questions that will divide the field into fairly equal parts, so eliminating as many incorrect answers as possible. Mathematically, carefully chosen questions allow the questioner to eliminate half the possible answers at each turn so identifying a single object from a field of 2^{20} or 1,048,576 within the twenty steps. As Charles Sanders Pierce said in 1901 “Thus twenty skilful hypotheses will ascertain what two hundred thousand stupid ones will fail to do.” This can usefully illustrate a binary search where an item is compared against the middle item in a sorted set; if the key is less than the middle item, the half which is greater is discarded and vice versa. This is repeated until the item is found or the set is empty. A decision tree, asking a ‘best question’ at each branch, so separating data groups represented by left and right children, might also support this kind of application well. However another useful algorithm could be the k-Nearest Neighbour, used in pattern recognition and machine learning.

20Q.net is based on a neural network and an algorithm. It is continually evolving based on input from users and learning by repetition. The game reports contradictions and allows changes to mistaken answers. It gives opportunities to introduce children to algorithms in a meaningful, relevant way, and to discuss how computers learn and evolve. *Lyndsay Hope*

WILL MACHINES EVER BECOME CREATIVE?

Issue 18 of the cs4fn magazine will explore whether machines can be creative. Ada Lovelace first suggested in the 1800s that one day they might, and now researchers are making it happen. We look at the first attempt at a creative algorithm (for writing love letters) to more recent programs that generate novel story lines. The machines are also creating music: from programs that evolve better music to ones using music to improve their relationships with humans. We even look at artificial intelligences trying to create new more powerful magic tricks. Whatever kind of art we may want to create, there are computers having a go at creating it.

Many teachers use cs4fn to support their teaching and as a way to keep up to date with the subject. If you don’t receive a copy make sure you visit cs4fn.org.uk and join the subscription list.



A new project from the cs4fn team: “A little bit of cs4fn” aims to bring the fun side of computing to younger kids. Do take a look at www.abitofcs4fn.org for short factoid versions of cs4fn. If the story interests you, then you can go on to a full cs4fn story, or you can go to another story in the same theme. Themes so far include superheros, spies, robots and AIs with much more to come.

Paul Curzon

COMPUTING AT SCHOOL

EDUCATE · ENGAGE · ENCOURAGE

Computing At School was born out of our excitement with the discipline, combined with a serious concern that students are being turned off computing by a combination of factors. **SWITCHEDON** is published each term. We welcome comments, suggestions and items for inclusion in future issues. Our goal is to put the fun back into computing at school. Will you help us? Send contributions to **newsletter@computingatschool.org.uk**

Many thanks to the following for help and information in this issue: *Dan Aldred, Phil Bagge, Irene Bell, Louise Branch, Neil Brown, Jon Chippindall, Andy Colley, Paul Curzon, Claire Davenport, Roger Davies, James Dent, Peter Donaldson, Mark Dorling, Tim Eaglestone, Sue Gray, Claire Griffiths, Lyndsay Hope, Marielena Ivory, Catriona Lambeth, Ian Lynch, Stephen Marchant, Peter Millican, Bill Mitchell, Bruce Nightingale, John Palmer, Emma Partridge, Rajal Pitroda, Chris Roffey, Matt Rogers, Linda Rowe, Sue Sentance, Andrew Shields, John Stout, Victoria Tatler, Dave White, Mike Wilkinson and Matt Wimpenny-Smith.*

www.computingatschool.org.uk

Computing At School are supported and endorsed by:

bcs The Chartered Institute for IT
Enabling the information society

Microsoft Research

Google

CPHC The Council of Professors and Heads of Computing