

COMPUTING AT SCHOOL

EDUCATE • ENGAGE • ENCOURAGE

In collaboration with BCS, The Chartered Institute for IT

SWITCHED ON

COMPUTING AT SCHOOL NEWSLETTER

SUMMER 2014



COUNTDOWN TO COMPUTING

This September, a new National Curriculum subject for all key stages comes to life: Computing. This historic change, the largest since the National Curriculum was conceived over twenty five years ago, reflects the growing awareness of the primacy of Computer Science. The concepts it expounds lie at the heart of the new subject. For many teachers much of the terminology is new, but stepping up to Computing need not be daunting. You are not alone. The longest journey starts with the first few steps, and CAS are gearing up to help you on your way.



Our Network Of Excellence continues to grow. At its heart are a multitude of local CAS hubs, providing us all with mutual support. Led by Computing enthusiasts, a warm welcome is assured. Hubs provide the glue that binds our community together - a place to share your concerns amongst friends. As we go to press, CAS have just received a generous grant from Microsoft through which we hope to double the number of hubs, amongst many other things. Further funding has been secured to develop the Barefoot Computing project to support primary schools. The Master Teacher programme, through which serving teachers can provide low cost CPD and advice, has attracted a large number of applicants and we are drawing these subject leaders into closer relationships with hugely supportive academic departments at over seventy universities.

Together we have a once in a lifetime chance to shape the emerging subject for the benefit of all our children. If you are confident, apply to be a Master Teacher. If your school is already 'walking the walk', become a Lead School and help those taking their first steps. Good teaching has always been based on collaboration and mutual respect. Helping others helps you develop too.

INSIDE THIS ISSUE

The summer term, and teachers everywhere turn their attention to planning for September. Curriculum design is the central theme running through this issue. It need not be daunting. CAS CPD co-ordinator, Mark Dorling kicks off by offering a framework around which you can start to plan.

Colleagues from around the UK, particularly primary teachers, share their advice and approaches to curriculum planning and rewarding achievement. The middle pages provide a suggestion for progression pathways which you can 'cut out and keep'.

In between you'll find features and reports of exciting things already being done by teachers at the cutting edge. We hope **SWITCHED ON** reflects the energy and enthusiasm of an emerging 'community of practice'. That community will be gathering for the sixth annual CAS Teacher Conference on June 21st. Join us for a day of talks, discussion, workshops and sharing. The atmosphere is infectious. Places are going fast, so register quickly. See the back page for details.



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.

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



Mark Dorling

APPROACHES TO DESIGNING A NEW COMPUTING CURRICULUM

Rome wasn't built in a day. Likewise, a new Computing curriculum will take time to evolve. There is no blueprint. Mark Dorling, the National CPD Co-ordinator for CAS, offers some advice about ways to approach your curriculum redesign.

CONCEPTS ARE KEY TO DEVELOPING THINKING

For a subject that has hardly existed in school, teachers have no shortage of material. At times the variety of software, and hardware can seem bewildering. There's a real temptation to try everything, buy everything and seek comfort under the blanket of resources, schemes and plans now available. Before you do, pause and consider what you are trying to achieve. It takes time to develop a scheme of work. Progressive refinement is the hallmark of a reflective practitioner. Small changes, implemented carefully, build confidence. Confidence breeds success.

Whatever projects you embark on, reflect on what you wish to instill. The key concepts, like handling complexity by decomposing tasks into smaller steps, or increasing efficiency by recognizing repeating patterns or simplifying solutions by abstracting away detail, allowing you to see the wood from the trees, emerge again and again, no matter what the context. Abstraction, decomposition and pattern recognition lie at the heart of computational thinking. When you try new things it is easy to get consumed by the practical detail, but it is computational concepts that provide the foundations for progression. Concepts allow children to make links and discover deeper meaning in topics. Delivering a curriculum is a means to an end. Children learn best through doing, but to learn to think like computer scientists they need a conceptual framework too. *Roger Davies*

The 2014 National Curriculum was published last September after 10 months of consultation. The two page, slimmed down curriculum is designed to provide teachers with greater flexibility and autonomy. However, combined with no statutory guidance on assessment, it presents a major challenge for teachers to correctly interpret the breadth and depth of learning, whilst ensuring a balance between the three strands identified: Computer Science, Information Technology and Digital Literacy.

My first advice for colleagues is "Don't panic". September 2014 is when the National Curriculum comes into affect which means that you do not need to have everything in place on the first day of the new academic year!

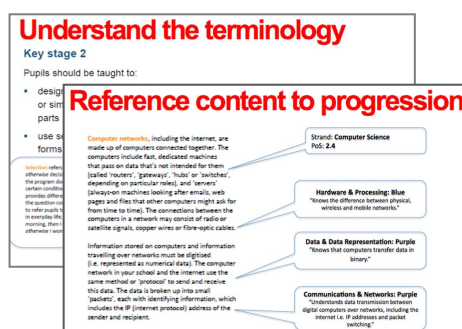
The new curriculum raises some interesting questions about pupil engagement. How do we enthuse and inspire the most disengaged and challenging pupils about Computing? And what about those with special educational needs?

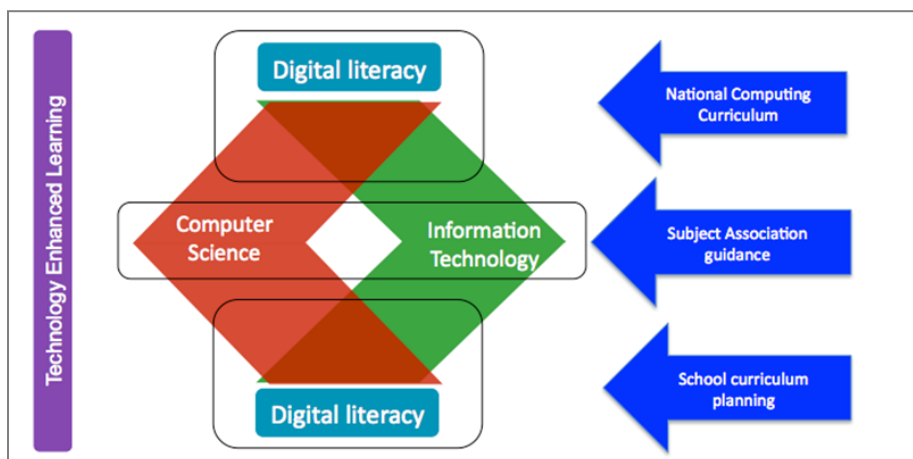
Marketing methodology says that people buy 'Why' and not 'What'. Yet traditional ICT curriculums that I have been guilty of planning in the past have focused on 'What' we were teaching, 'How' we were going to teach it, and only finally 'Why' we were learning it.

Engaging all pupils, but particularly girls, can be achieved by listening to what interests them. Perhaps focus on real life and/or cross curriculum challenges (the Why), then stress using Computational Thinking and the principles and concepts behind it to solve the problem (the How) above the product they produce (What). This also helps to address the expectations and perceptions of pupils that computers are a tool for learning not the focus of the learning.

With this in mind, begin this transition with your colleagues by examining any misconceptions within your teams about the National Curriculum and work with colleagues to build a department or school vision for Computing that meets pupils' needs. Encourage all colleagues to define "curriculum", "innovation" and "creativity" in light of their expertise and specialisms, with a view to combining these definitions into a single vision or statement. Use this to create a 1, 2 and 3 year plan – with a range of activities that tie in with the vision statement identified by you and your colleagues.

Computing At School members and the wider community (via CAS Online) have released a range of guidance documents (non-statutory) to support teachers. For example CAS, Naace and ITTE produced a joint statement, CAS produced Computing: A Curriculum for Schools, Primary and Secondary Guidance, and there are various Assessment Frameworks and discussions around Digital Badges, all available on the CAS Community. In the middle of this issue you'll find more suggestions to stimulate your own ideas in our special focus. You can use this non-statutory documentation to help interpret the study require-





ments in the National Curriculum, both to understand the vocabulary and show the breadth and depth of what is to be taught. As you set about interpreting the programme of study, you'll find it raises questions about learning pathways and progression looks like.

You can consider progression in the National Curriculum by mapping the CAS Guidance for your phase and National Curriculum to the Assessment Framework of your choice. This process will help you to order and group concepts and principles, and through that you can't help but notice the dependencies and co-dependencies between skills, concepts and principles.

A good starting point for your curriculum design might be to first evaluate your existing units of study and map them to the National Curriculum and your chosen Assessment Framework. Then consider the transitional challenges and strategies available to you for effective support between Key Stages Two and Three, and KS3 and KS4, both in terms of recognising the lack of prior learning and the overlap

between KS3 and KS4 qualifications.

You can use this mapping exercise to consider what areas of the curriculum you should prioritise for your CPD needs. You can identify the areas of the curriculum where you are able to refocus existing units. For example, in lessons looking at e-safety and smart searching, you could also consider investigating network concepts such as IP Addresses and Packet Switching. Again, there are many data representation activities produced by CS Unplugged that could enhance existing photo editing or graphical projects. Other areas may require adopting and developing new units of study whether that be from something shared by the community or purchased from a commercial provider.

In many ways, you are spoilt for choice. It is a nice problem to have, but easy to feel overwhelmed. The key is to implement gradual changes, ensuring you can walk before you try running. Curriculum development is an ongoing process, best shaped by reflecting on what you have learnt as you develop and trial new approaches.



Funded by the DfE to help primary school teachers prepare for the new computing curriculum, there are three main aspects to the Barefoot project: creating materials; running workshops; and developing a community of practice.

Exemplar Teaching Materials:

Developed by computing teachers Jon Chippindall, Zoe Ross and Jane Waite, and an editorial team of Miles Berry and John Woollard, high quality primary resources are currently being created to help teachers deliver the curriculum from September. The activities cover a wide range of cross-curricular situations and demonstrate how subjects such as English, Maths, Science or History can benefit from the new computing curriculum.

Teach Yourself Notes: A series of self-help notes designed to support primary teachers on their journey towards becoming excellent computing teachers. Covering key concepts, language and vocabulary, they will allow them to teach with increased confidence and understanding.

Barefoot Workshops: Starting in summer, free CPD sessions run by volunteer experts will introduce the Barefoot resources.

Barefoot Communities: Launching in September to share ideas and good practice with other primary teachers. Barefoot is supported by many different organisations. The project is managed by Pat Hughes. Find out more at <http://barefootcas.org.uk>, mail info@BarefootCAS.org.uk or follow @BarefootComp for updates.

BRINGING TEACHERS TOGETHER AT PRIMARY CONFERENCES

A series of CAS regional conferences aimed at Primary teachers were taking place as SwitchedOn was being prepared. The North East conference, hosted by Newcastle University was attended by nearly 170 teachers. Next up is the North West, hosted by the University of Cumbria. They will be offering events at both their Carlisle (Monday June 9th) and Lancaster (Friday June 13th) sites. Early bird registration is only £55. A day of high quality CPD delivered by experienced teacher trainers and primary practitioners is planned. Practical sessions specific to KS1 and KS2 will identify cross-curriculum links and cover a range of topics. The aim of these one day conferences is to support primary teachers with the introduction of the new Computing curriculum. If there hasn't been one near you, why not organise one?

HAVE YOU CONSIDERED BEING A PRIMARY MASTER TEACHER?

The role of the CAS Master Teacher is to build capability in computer science teaching by running not-for-profit CPD sessions, perhaps in association with local universities and other partners, and providing an informal consultancy service for teachers in their local area. We are particularly keen to hear from primary teachers or teachers who are involved in cross phase work.

If you can answer 'YES' to the following then the CAS Master Teacher programme is for you!

- Would you like funding to develop your skills and knowledge of Computing in the new National Curriculum?
- Are you a primary teacher in a state maintained School?
- Are you already expert in delivering the 1999 ICT National Curriculum?
- Are you an experienced teacher with a judgment of good or outstanding teaching seen in your recent lesson observations?
- Can you confidently engage with your peers in a professional environment?
- Do you have the ability to design and deliver practical and interactive workshops for teachers with appropriate course material?
- Do you have a passion about sharing best practice in teaching and learning?

If you are keen to see Computing develop, are active in your local hub and would like your school to receive funding to release you to help others in the locality, becoming a Master Teacher could be the answer. As well as Level 2 Master Teachers, who already have experience, we are also interested in hearing from potential Level 1 Master Teachers, willing to undergo training. For further information on the programme, please download the information pack (bit.ly/1hmqvVx) or email the CAS Network of Excellence team: noe@computingatschool.org.uk. We look forward to hearing from you.



APPS TO USE IN THE PRIMARY CLASSROOM

Andrew Shields is a reception teacher in a village school in Leicestershire, delivering training on tools to use in the new primary curriculum. Here he outlines some of the options available.

There has been a lot of discussion on the CAS website as to which tools would be best to use when looking at the computing aspects of the new curriculum and also whether iPads and / or other tablets, have anything of use to give in this area. I think they all have something to offer and where possible children should have access to a range of experiences. As with a lot of things to do with computing, it all depends on your kit, budget, time and skills.

As the weeks go by, there appear to be more and more options available to us. There are programs that are Flash-based and so are not suitable for all platforms; there are apps that have just been written for the iPad; a smaller number that have been developed for the Android platform and an even smaller number that have been developed for the Windows platform. There are very few programs that will run across all platforms. For Key Stage 1 there are the following you could try:

- Daisy the Dinosaur (iPad)
- BeeBot (iPad)
- A.L.E.X. (iPad)
- Kodable and Kodable Pro (iPad)
- Cato's Hike (iPad)
- Light-bot (iPad)
- Robologic LE (iPad)
- Hopscotch (iPad)
- Scratch Jr (as of time of writing this has not been released)



For Key Stage 2 you can add:

- Scratch 2.0 (web based Flash and downloadable)
- Kodu (a Microsoft download for PC and Xbox)
- Tynker (a web based program that can be used on all platforms)
- DynamicArt Lite (iPad)



There are also a number of web-sites that will take you through programming such as codecademy.com, code.org and Code Club (see page 7). You can get yourself some Raspberry Pis and head that way too. The above lists are not exhaustive. There are other things you may be aware of that you could try and each month there seems to be something new as we rush towards September. The above list is also heavily weighted towards the iPad; this just seems to be where current development is. Don't forget your BeeBots, Pixies and Roamers either. These all help to fulfil the computing requirements.

Also, please bear in mind that there are a number of aspects in this curriculum that can be done, and work very well, away from the computer. You may have been doing a lot of the new curriculum under another name; for example, in literacy writing a set of instructions (algorithm) for the often referred to 'Jam Sandwich'; in science when talking about the things you need to grow a bean and then when it hasn't grown, working out why (debugging). Happy programming!

PROGRAMMING AT KEY STAGE 1 USING BIGTRAK

The Jurassic Coast Teaching Schools Alliance held a series of twilight sessions for primary teachers. Organiser Adam Shelley, Assistant Headteacher at the Woodroffe School in Lyme Regis outlines the lessons learned.



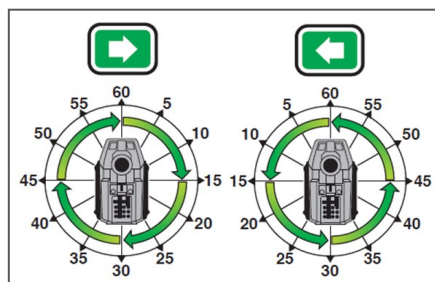
So why choose Bigtrak to start students programming? The answer is simple: it is what actually got me into programming in the 80s at primary school. The main thing I remember from my early primary days was sitting on a carpet programming this machine to go around objects and the fantastic noises the machine made when it moved. I remember writing down instructions in a sequence, testing them and then changing them to make it move around the objects. This is exactly what the new primary curriculum requires. You can get a Bigtrak for around £30 (£20 for a Bigtrak Junior). This is cheaper than other floor turtles so I ordered two and set up the course. The resources are on the CAS Community (bit.ly/1hmwGZN).

I had arranged various tasks around a school hall, included a line with equally spaced cones which Bigtrak had to weave in and out and a shape (made out of masking tape) to go around but you can design any courses you want or get the students to design them. We ended with a race of the Bigtrak around an obstacle course which really brought out the competitive nature of teachers and I am sure would do the same with students!



Students need to know how Bigtrak uses distances and turns before they can start to develop their algorithms. The key concept is that every 1 move command is equivalent to 1

Bigtrak length. As this is about the size of a ruler or A4 landscape sheet of paper these can be given to students to help estimate move commands. Turns prove more difficult but we simply used the diagrams shown printed on acetate to estimate the turn required. For younger students the right turn might be the only one you use as it can be thought of as a clock face. You can have a lot of fun by children acting out instructions which will help fix the concept.



Modelling actual movement helps them develop the algorithms they need. The key thing is getting resources simple enough to use with reception students. The most effective resource allows students to place the numbers and arrows on a sheet. This takes away the need to write instructions and allows them to move them around when testing and debugging. Once teachers had got the concepts we focussed on how we might use Bigtrak into KS2 by making algorithms



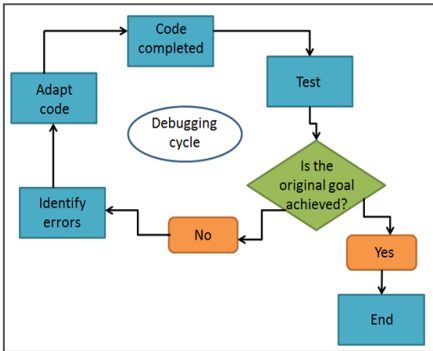
more efficient. This meant that we could use the repeat button, thinking in a more computational manner to solve more complex problems. Children can write, rather than arrange instructions, progressing to using the fewest steps possible. This can lead on to predicting what will happen with a focus on logical reasoning. By the end of the course the teachers had the skills not only to use Bigtrak but also see what some of the new objectives in the curriculum actually mean. Through explanation and the modelling done with Bigtrak they realised they're already doing a lot of the work as part of control teaching in the current ICT curriculum. They now need to adjust the language to ensure students know they are developing, for example, algorithms.



The concepts here are not Bigtrak specific. If you have any floor turtle you can adapt the resources to fit. Even without devices you can introduce the concepts by getting the kids to be the Bigtrak. They move around following the algorithms that you or others have developed. Moving from ICT to Computing should not be costly. Use what you have or invest in a couple of Bigtrak to use with groups of students. The main thing is that your children have fun while developing their computational thinking! Who knows, in 30 years' time they might remember using Bigtrak at the start of their programming career like I did!

DEVELOPING THE CURRICULUM IN THE **KEY STAGE 1** CLASSROOM

The new Computing content may not be as difficult as it first appears. Emma Goto, a former AST for ICT, now lecturing in Initial Teacher Training at the University of Winchester, reassures KS1 practitioners that Computing is a lot easier than they may think.



Much of the programming content of the new KS1 curriculum may not be as difficult as it first appears. In most cases it will build on the current provision in schools. What about the rest of the subject content?

The National Curriculum states that children “should be taught to ... use technology purposefully to create, organise, store, manipulate and retrieve digital content”. Talking books, podcasts of class songs, videos of special events, mindmaps to organise thinking, graphs to present data collected in maths, posters linked to their work—the list is endless.

Children have always enjoyed using ICT to create a range of digital content. I would expect all these rich, creative and motivational uses to be carried forward.

Finally, children’s use of online resources has developed rapidly, opening up lots of positive opportunities. But there are some risks that run alongside that. Knowing how to stay safe and behave well is crucial for children growing up today. The requirement to teach children to use technology safely and respectfully is a welcome development that simply reflects these changes in society.

With any change, the unknown can cause a level of anxiety but once we move forward we often realise things are not that different. Words such as algorithm, coding and debugging will be new to many teachers. However, when they are unpicked many KS1 teachers will be reassured to find that they may already be teaching much of the new Computing curriculum.

Young children love learning new words and technical terms. When teaching children as young as four and five years old we use words like digraph, phoneme, grapheme, symmetry and reflection. These words quickly become part of their vocabulary. The language of computing will soon be absorbed by children as they describe concepts that, in many cases, they are already using.

The KS1 children that I have taught have loved working with programmable toys, such as BeeBots and Roamers. Initially, they were given lots of opportunities to play with and explore the devices. Through trial and error they developed an understanding of how to use the buttons to program the device. As their learning progressed they were set the task of moving the device around maps and grids in order to achieve a given outcome. Most of the children were excited by the opportunity to work in small groups, collaborating to solve a problem. The problems were usually linked to current themes and topics to give them a real and meaningful context. These were rich challenges which provided lots of fun for us all!

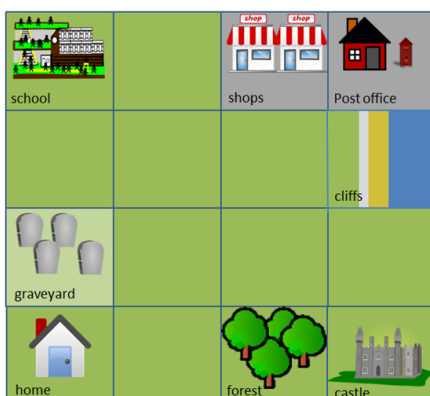
On the grid shown, children might be asked to find the shortest route to move the toy from its home to the Post Office, in order to collect the

mail, then, deliver it to the castle, whilst avoiding the graveyard, the forest and the cliffs. In solving these problems children planned out the sequence of steps they would need to go through. They would record their plan in their own ways, using words, symbols and pictures.

An algorithm is basically a logical sequence of steps to solve a problem - therefore the children had written an algorithm! They then used the buttons on the programmable toy to input the instructions that would be needed to get the device to work through the algorithm - they were coding! They used the buttons to create code in the programming language of their toy.

Whenever they had programmed the device in this way they would press go and execute their program. Children were then encouraged to evaluate their program. Had it achieved what they had set out to do? If not, why not? What element of their program did they need to go back and change, in order to ensure they had solved the original problem? They went back to their program and altered it. The children continued through this cycle of executing the program; testing and observing what happens; identifying problems and fixing them, until their program worked. This process, which is illustrated in the diagram (top left), is referred to as debugging by programmers: a word children will soon also be using.

The sidebar considers other elements in the new curriculum. The move from ICT to Computing in Key Stage One is a challenge, but for those that were providing a broad ICT curriculum the change may not be as great as it first appears!



LEARNING TO CODE IN AN AFTER SCHOOL CLUB

Kevin McLaughlin, from Old Mill Primary School in Leicestershire shares his experiences of running an after school Computing club and sings the praises of code.org



When I started an after school Computing club it was oversubscribed so the first 20 children got in. It's not all boys either. There are 6 girls in the club and a few more waiting their chance to join later in the year. To say it got off to a great start would be a slight understatement - feedback from the young 'coders' and their parents has been excellent.

I decided to stay away from a 'Learn to use Scratch' club as the same children will be using Scratch next year as part of the Computing Curriculum. From various possibilities online I found that learn.code.org/ would be perfect for my needs. Part of code.org, it has an Introduction to Computer Science course that runs for 15-20 hours. It is broken into 20 stages. Each stage has mini activities the children complete to gain your awards.

Once registered as a teacher, you can easily set up your students. Add them yourself or get them to do so with a unique 6 character code. This is perfect for schools that don't have access to email or prefer not to use it to sign up for online activities. Each user has their own progress board that details their completed stages and what they have left to do.

After the introductory 'What is a computer scientist?' video, the children

started Stage 2 and were hooked straight away. They used a Scratch like 'drag and drop' interface to move a character around a maze - based on an Angry Bird catching a pig. As they progressed through the activities, they shared solutions, working together to solve the problems they were presented with. They looked carefully when they discovered their 'code' didn't quite function in the way they thought it would and I encouraged them to go through each step to debug it. Cheers went up when the first child won a trophy and others flocked to discover how he had solved the puzzle.

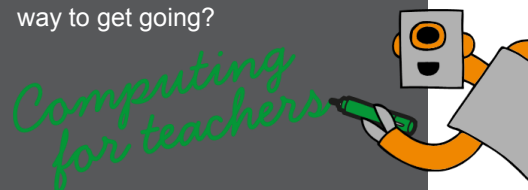
An excellent addition to the teacher section is the ability to view the progress of your class but also run the code that they have been working on. I can add a comment so the child can develop a better understanding of how to improve their code if it is not working correctly. The child can click 'See code' and see how their moveable blocks are actually represented in JavaScript.

The following weeks saw the children progress through the stages at their own pace - some getting further along and so being able to help the others. Learn.code.org is a fantastic site, well thought out and very simple to get you and your school started in its coding journey. I thoroughly recommend it.

CODE CLUB TO TRAIN TEACHERS

{ CODE
CLUB }

If you're a primary school teacher and unsure about how to get started, a good place might be Code Club. Launched two years ago, Code Club has marshalled a massive volunteer army of IT professionals to run after school Code Clubs for primary children alongside their class teachers. The scale of growth is a measure of the goodwill that exists to help ensure the new curriculum is a success. If you would like to make contact with a volunteer, simply visit the Code Club website and post a request. Likewise, if you are an IT pro, willing to help organise a club, simply sign up and Code Club will be in touch. You won't be alone - there are nearly 2,000 Code Clubs now running, supported by materials developed for everyone to use. What better way to get going?



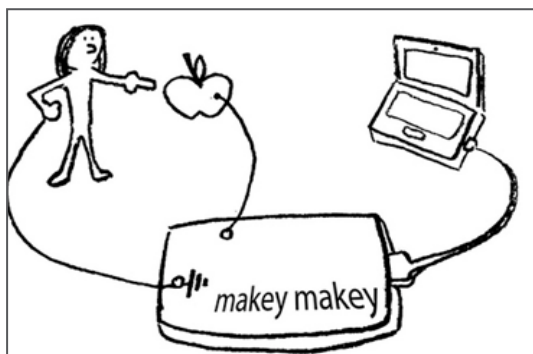
A partnership between CAS, Google and Code Club saw the launch of Code Club Pro in February. Having got over 28,000 children started with coding in the UK, Code Club will now deliver Continuing Professional Development (CPD) training to primary school teachers. The materials have been created in collaboration with Computing At School and primary school teachers across the UK. The sessions aim to demystify the new curriculum, explain key language and concepts, and provide useful materials to use in and out of the classroom.

They won't be boring either. Just like Code Clubs, the emphasis will be to learn through doing and having fun. To find out more and sign up for notification when there is training in your region visit <http://codeclubpro.org/>

K-8 Intro to Computer Science Course (15-25 hours)

This 20-hour course introduces core computer science and programming concepts. The course is designed for use in classrooms for grades K-8, but it is fun to learn at all ages.





BRINGING INPUT HARDWARE TO LIFE WITH **MAKEY MAKEY**

Ever thought about using fruit or vegetables as inputs to control computer programs? Andrew Daykin, a teacher of ICT, Computing and Media at Heysham High School in Morecambe introduces the amazing Makey Makey.

MAKEY MAKEY: THE **GEEKY STUFF AND GREAT IDEAS**

Makey Makey is a printed circuit board with an ATmega32u4 micro-controller running Arduino Leonardo firmware. It uses Human Interface Device (HID) protocol to communicate with your computer. It can

send key-presses, mouse clicks and movements by creating circuits through materials such as people, fruit, vegetables and play-doh. The Makey Makey was invented by Jay Silver

and Eric Rosenbaum in conjunction with the MIT Media Lab. More to read at: <http://www.makeymakey.com> and <http://scratch.mit.edu>

The Beetle Scoffer game is available to remix on the Scratch website. Here's a few other ideas that have been fun:

- Carrots replace the arrow keys for a rabbit based game.
- High five the head teacher. Simple skin on skin contact to produce a random result, from a free GCSE to detention!
- Banana keyboard. Several scratchers have made keyboards and drum machines that are playable with the fruit and veg of your choice. Other online musical apps can also be played.

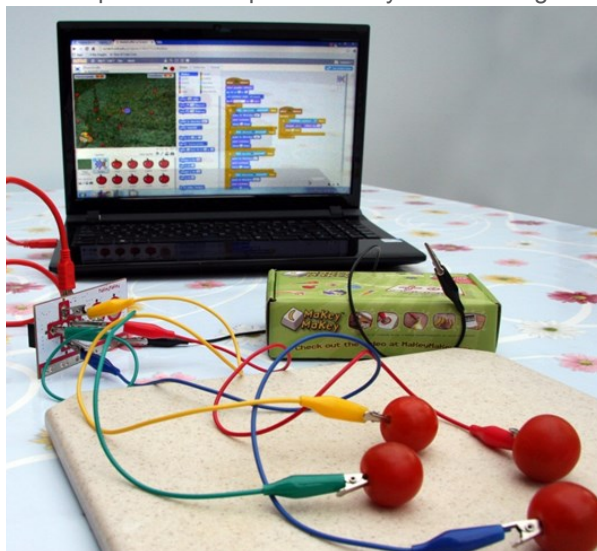
Try anything a little conductive. The best advice - experiment.

One constant challenge that we face when delivering demonstrations and lessons about computer hardware is how we can maintain interest and engagement when most of the devices are so familiar that young people almost disregard their technical details. A mouse simply does what it does and we use it without regard, much as we take our fingers and thumbs for granted.

Whilst planning a series of lessons about input and output devices a few years ago I came across a video on YouTube which demonstrated the magical things you can achieve with a Makey Makey. I was so impressed I went to the inventor's US website and ordered one straight away. Three months later I received my package from the States. By this time they had become available on next day order from UK sellers and I had already equipped myself with a second purchase. But what could I do with it? The Makey Makey is a small circuit board that offers the user the possibility of inventing their own means of interacting with apps and games by imagining, experimenting and creating replacements for the keyboard and mouse. That's all really great, I hear you say, but what did you do with it?

My first opportunity to use my new toy coincided with an open evening. I had just discovered Scratch 2.0 and decided to make the demonstration more exciting by linking a basic game in which a beetle was eating cherries by making a controller out of cherries.

How did it work? First I could not get hold of cherries... so I connected a cherry tomato to the up, down, left and right controllers and placed them on the desk for the primary school visitors to control the game. Each player must also be connected to the earth lead so a friend held the earth and completed the circuit by holding the players ear. This caused much amusement and during the game play I explained how the device made it possible to replace the keyboard. A league table soon appeared



and youngsters queued up to try the game. Some even ate the tomatoes when they finished. We also experimented with hand drawn game controllers simply using pencil and paper. This kept the waiting, would-be players busy. See the sidebar for other ideas.

DESIGN, MAKE AND THEN PLAY 'REAL' ARCADE GAMES

Arcade games provide a fertile context for school projects. But how can students then play them? Terry Watts, and Sam Lawrence from Cotham School, Bristol outline their unique approach.



Back in March 2013 I started an after school programming club. It was started after we delivered a successful scheme of work in GameMaker to Year 8. We started the club with 13 students. They used the club to extend their programming skills by developing more advanced games using extra tutorials we found on the internet. Students progressed from drag and drop programming to the built in scripting language. After a few weeks we began talking about how we could let people play the games we were creating. This spawned the idea of creating our own school arcade cabinet which could contain all our games. I loved the idea but we needed help if we going to do it.

I took the idea to Sam Lawrence, one of our DT teachers. He liked it and immediately produced some rough sketches of what it could look like. After seeing the sketches we were hooked. Production of the cabinet began on Wednesdays and development of the games continued on Fridays, both after school. The school's enrichment fund paid for the wood and

monitor. Within a few weeks the students had the two sides of the cabinet cut out and we were sizing up how the monitors and computer would fit inside. By October we'd finally got the cabinet together. We ordered in proper joysticks from an arcade cabinet supplier. Fitting these suddenly made the cabinet feel like the real deal. We learnt a lot about the electronics inside the cabinet during this time as we had to learn how to wire the joysticks in and how their inputs would be translated into keyboard presses.

As the cabinet reached its final stages we had to think about the control systems for our games. We had to ensure that each game used the same control system to ensure they worked with joystick inputs. We also had to program in a 'home' button that allowed users to exit the current game and go back to the game select screen. This meant a shared agreement on how this should work, so everybody could program the feature in the same way.

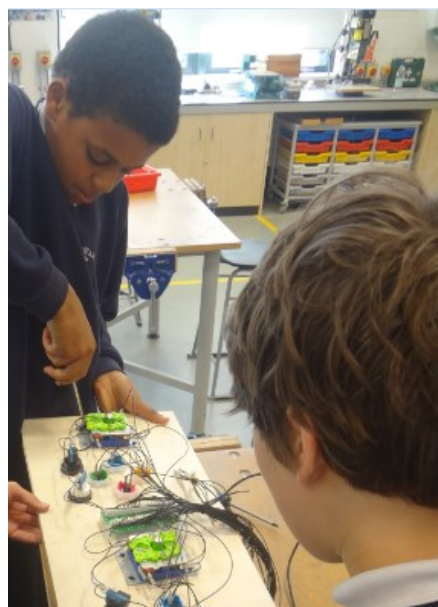
The artwork for the cabinet was put together by the school's graphic designer. The logo for our games club was designed by a Year 8 girl who was an amazing artist. The last week of production was frantic as we worked to make our games as good as possible whilst bug testing on the arcade cabinet. The frontend game select screen was programmed by a student. He programmed it in C#, which he taught himself. After 10 months of work the cabinet was finally launched at Digimakers, a technology event hosted by Bristol University and BCS: The Chartered Institute For IT. The GameMaker students were on hand to explain the games and describe how the cabinet was made. The feedback from all who saw the ma-



chine was amazing; they were all in awe of what the students had managed to achieve. It was a fabulous endorsement of their efforts.

The whole project has been an amazing learning experience for all of us, students and teachers. I am so proud of the students, they really did put the work in and the school has a great arcade cabinet as a reward.

Mr Lawrence and I already have our next project planned and one that will reach out to some more potential programmers and engineers within our school. Check out the full project blog at our Computing website: <http://bit.ly/1m45PtJ>



FINDING AND SHARING INSPIRING STORIES

Have you done something in your school a little out of the ordinary? Have you had your children coming back asking for more? Has something worked that others could benefit from? Whether it is a big project or small insight **SWITCHEDON** wants to share your success. CAS is a community of practice that strives for excellence in Computer Science teaching. If you can contribute an idea or article please get in touch. newsletter@computingschool.org.uk

PROGRESSION PATHWAYS FOR COMPUTING

Despite the opportunities for autonomy with no statutory assessment framework, school systems will take time to adapt. It will also take time to develop new ways to measure progress. There is also the immediate challenge to get pupils up to speed with the prior learning for the Key Stage.

The Progression Pathways (centre pages) is one example of a non-statutory Assessment Framework to support teachers assessing pupils' progress. It is one interpretation of the breadth and depth of content in the National Curriculum. The focus of the framework is progression through and across the strands, rewarding pupil achievement via coloured digital badges.

The progression through each strand is broken down into rows. The rows are colour coded (like karate belts) to help the teacher assess student competence at different levels. If you plan to use the framework with your existing assessment/reporting system then you can assign arbitrary levels to the coloured rows. Agree the benchmark 'level' for the pupils entering a particular key stage and assign these to the appropriate progression statements (colour) for each strand.

It is suggested that eventually Primary teachers focus on the badge statements from the Pink to Blue rows and Secondary teachers focus on the badge statements from Purple to Black. The white row overlaps with the KS4 specifications. If pupils are working between two colours (rows) for a particular badge, then we suggest a two-tone badge is available so you can record and reward sub levels of progress. There are no Digital Badge designs for the column headers. The authors of the document believe teachers who would be using a Digital Badge system would be much better placed to design and create them, engendering greater ownership by pupils.

ADAPTING YOUR CURRICULUM ONE STEP AT A TIME

If preparing for September still seems daunting, Taryn Hauritz, from Thomas's London Day Schools, offers her advice on rewriting a whole school curriculum.



The new computing programme of study states that "A high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world". Wow! This is pretty exciting stuff. But, if you're anything like me and don't have a background in computer science, where on earth do you begin?

A few weeks ago, I started to write a new computing curriculum for Thomas's London Day Schools. Having taught ICT there for 10 years, I thought that this would be a fairly straight forward process and I was excited about the possibilities. But, after several weeks of researching and studying, I found myself going round in circles. It was almost like there was too much information, but somehow not exactly what I was looking for. So, after extensive networking, tweeting and compiling of many, many ideas I'd like to share with you what I've learnt.

STEP 1: Read "Computing in the national curriculum: A guide for primary teachers". This is price-less. It explains everything you need to know to get started.

STEP 2: Download the Computing Progression Pathways. Familiarise yourself with each of the six strands. It's important to note that primary children are expected to reach the end of the blue strand.

STEP 3: Get out your current ICT curriculum and use the six strands above to identify any areas of weakness in your current scheme of work. I actually found it easiest to download the child-friendly version (see step 7) and stick the descriptors on my yearly overview.

STEP 4: Use the CAS website to find resources (including CPD) to help you plan any new units of work to "fill in the gaps". If you need more time to do this properly, leave teaching these units to next summer.

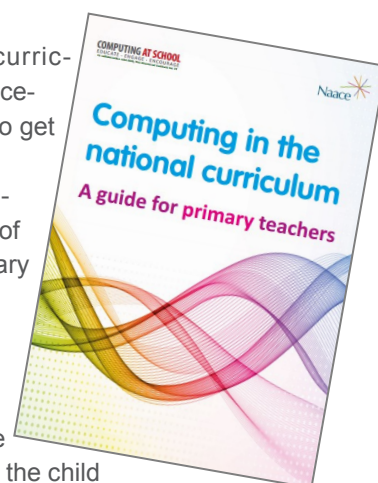
STEP 5: Adapt the units of work you'd like to keep for the new curriculum by updating your medium term plans with the new Programme of Study and the Progression Pathways indicators in step 2.

STEP 6: Add Computer Science Unplugged activities to as many of your units of work as possible. They really help encourage computational thinking across the curriculum and are great for kinaesthetic learners.

STEP 7: Download the child-friendly computing progression pathway statements from the CAS website and use them for pupils' self and peer assessment. I would also use them for an interactive display so that you and your pupils can reference them regularly.

My final advice to you is to "Reach Out". Keep networking and discussing ideas with as many colleagues as possible - a problem shared is a problem halved and, together, we really can change the world! For more detailed tips and links to all of the resources above, go to

www.primaryschoolcomputing.co.uk



DEVELOPING BADGE MISSIONS TO CELEBRATE PUPIL PROGRESS

As part of their curriculum planning, teachers are exploring how best to celebrate pupil progress. Matt Rogers, Computing subject leader at Snowfields Primary School in Southwark outlines their approach to developing digital badges.

One area in which we felt there was currently a distinct lack of resources and support for schools relates to assessment. Recently we have been working with Makewav.es, the social networking platform for schools, as a home for badging, which is linked to Open badges, DigitalMe and BadgetheUK. In partnership with Makewav.es I have been developing badges based around computing and coding. However this is only a section of the work we do on badging. The plan is, over time, to create a multitude of 'badge missions' (as they are called) for the wider curriculum, linking work to ICT throughout. An example of the existing badges that have been created by a community can be seen in Makewav.es Public Library at www.makewav.es/badgelibrary.

Although the badges themselves are not linked to any understood 'levels' within school, we created a level 1, 2 and 3 set of badges, which increase in difficulty and 'commitment' required from the children to earn them. A prime example of this can be seen through our 'Imagine, Program, Share' badges.

The designs for these particular badges were done by Makewav.es on our behalf, but for some badges we have used our connections with varying organisations to get them to design badges for us. For example Mathletics/3P Learning and 2Simple have helped out with images for badges related to missions I have created. Also available is the Makebadg.es structure to create original designs.



This option is often used by schools and pupils who are developing badges themselves. A really exciting opportunity at the moment involves the pupils themselves developing their own Badge Missions – the best of which are going to be created as 'actual' badge missions shared with the wider Makewav.es community. For me, this gives the power back to the children, allowing them to have a stake in their own learning, and ensuring that we, as practitioners, are teaching and learning alongside the children's needs and passions.

The badges themselves are awarded digitally, through the Makewav.es platform (with the option of the children carrying these over to their Open Badges backpack at a later stage). Teachers registered to the school site at Makewav.es can then award the badge once each mission has been completed. This, in turn, is then displayed on the pupil's personal page, so they can show off their achievements to peers and parents.

I would say the introduction of badges has been the single most important change in the way we have celebrated achievement at Snowfields, and has really driven the children to try something new. For example, if they see a badge their peers have attained it has the Mexican wave effect of making them take the badge mission and then their peers take it and so forth.


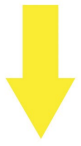






MORE UNPLUGGED TEACHING RESOURCES

Teaching London Computing (teachinglondoncomputing.org), aims to support computing teachers in preparing for the new curriculum, providing CPD courses and engaging new 'unplugged' classroom activities. Funded by the Mayor of London and the DfE with additional support from Google, it is a sister project to cs4fn, bringing together teams from Queen Mary University of London and King's College London. We provide subsidized CPD courses in various formats for teachers in London. Of course, those outside London may find other courses local to them through the CAS network. Ultimately we hope others will also run courses based on our material.

All our activities and booklets, covering topics from computational thinking to unplugged ways to teach programming, are free to download. Our latest booklet, joint with cs4fn, "Computational Thinking: Searching to Speak" explains what computational thinking problem solving is about. It uses the problem of helping someone who is totally paralyzed and has linked activity sheets. These materials are augmented with free workshops from project co-director Paul Curzon to demonstrate the activities, giving teachers a chance to see them first hand before taking them back to their classrooms. We ran the first set of workshops at QMUL but are keen to offer them from other London locations, so if you can gather a reasonable number of computing teachers together, get in touch and we'll do the rest.

Paul Curzon & Jo Brodie



Pupil Progression	Algorithms	Programming & Development	Data & Data Representation
	Understands what an algorithm is and is able to express simple linear (non-branching) algorithms symbolically. Understands that computers need precise instructions. Demonstrates care and precision to avoid errors.	Knows that users can develop their own programs, and can demonstrate this by creating a simple program in an environment that does not rely on text e.g. programmable robots etc. Executes, checks and changes programs. Understands that programs execute by following precise instructions.	Recognises that digital content can be represented in many forms. Distinguishes between some of these forms and considers the different ways that they communicate information.
	Understands that algorithms are implemented on digital devices as programs. Designs simple algorithms using loops, and selection i.e. if statements. Uses logical reasoning to predict outcomes. Detects and corrects errors i.e. debugging, in algorithms.	Uses arithmetic operators, if statements, and loops, within programs. Uses logical reasoning to predict the behaviour of programs. Detects and corrects simple semantic errors i.e. debugging, in programs.	Recognises different types of data: text, number. Appreciates that programs work with different types of data. Recognises that data can be structured in tables to make it useful.
	Designs solutions (algorithms) that use repetition and two-way selection i.e. if, then and else. Uses diagrams to express solutions. Uses logical reasoning to predict outputs, showing an awareness of inputs.	Creates programs that implement algorithms to achieve given goals. Declares and assigns variables. Uses post-tested loop e.g. 'until', and a sequence of selection statements in programs, including an if, then and else statement.	Understands the difference between structured and unstructured information. Knows why sorting data in a file can improve searching for information. Uses filters or can perform single criteria searches for information.
	Shows an awareness of tasks best completed by humans or computers. Designs solutions by decomposing a problem and creates a sub-solution for each of these parts (decomposition). Recognises that different solutions exist for the same problem.	Understands the difference between, and appropriately uses if and if, then and else statements. Uses a variable and relational operators within a loop to govern termination. Designs, writes and debugs modular programs using procedures. Knows that a procedure can be used to hide the detail with sub-solution (procedural abstraction).	Performs more complex searches for information e.g. using Boolean and relational operators. Analyses and evaluates complex information, and recognises that poor data leads to unreliable results, and draws conclusions.
	Understands that iteration is the repetition of a process such as a loop. Recognises that different algorithms exist for the same problem. Represents solutions using a structured notation. Can identify similarities and differences in situations and can use these to solve problems (pattern recognition).	Understands that programming bridges the gap between algorithmic solutions and computers. Has practical experience of a high-level textual language, including using standard libraries when programming. Uses a range of operators and expressions e.g. Boolean, and applies them in the context of program control. Selects the appropriate data types.	Knows that digital computers use binary to represent all data. Understands how patterns represent numbers and information. Knows that computers transfer data. Understands the relationship between data and file size (uncompressed). Defines data types: real numbers and Boolean. Queries data on one table using a typical query language.
	Understands a recursive solution to a problem repeatedly applies the same solution to smaller instances of the problem. Recognises that some problems share the same characteristics and use the same algorithm to solve both (generalisation). Understands the notion of performance for algorithms and appreciates that some algorithms have different performance characteristics for the same task.	Uses nested selection statements. Appreciates the need for, and writes, custom functions including use of parameters. Knows the difference between, and uses appropriately, procedures and functions. Understands and uses negation with operators. Uses and manipulates one dimensional data structures. Detects and corrects syntactical errors.	Understands how numbers, images and character sets use the same bit patterns e.g. binary addition. Understands the relationship between resolution and colour depth, including the effect of compression. Distinguishes between data used in a program (a variable) and the storage for that data.
	Recognises that the design of an algorithm is distinct from its expression in a programming language (which will depend on the programming constructs available). Evaluates the effectiveness of algorithms and models for similar problems. Recognises where information can be filtered out in generalizing problem solutions (abstraction). Uses logical reasoning to explain how an algorithm works. Represents algorithms using structured language.	Appreciates the effect of the scope of a variable e.g. a local variable can't be accessed from outside its function. Understands and applies parameter passing. Understands the difference between, and uses, both pre-tested e.g. 'while', and post-tested e.g. 'until' loops. Applies a modular approach to error detection and correction.	Knows the relationship between data representation and data quality. Understands the relationship between binary and hexadecimal circuits, including Boolean logic. Understands how and why values are data types. Understands different languages when manipulating programs.
	Designs a solution to a problem that depends on solutions to smaller instances of the same problem (recursion). Understands that some problems cannot be solved computationally.	Designs and writes nested modular programs that enforce reusability utilising sub-routines wherever possible. Understands the difference between 'While' loop and 'For' loop, which uses a loop counter. Understands and uses two dimensional data structures.	Performs operations using bit patterns. Understands conversion between binary and hexadecimal. Understands binary subtraction etc. Understands the need for data compression. Performs simple compression methods. Understands what a relational database is, and understands the benefits of storing data in multiple tables.

Progression Pathways

Statement	Hardware & Processing	Communication & Networks	Information Technology
Understands the difference between hardware and software. Can explain the difference between hardware and software. Can explain the difference between hardware and software.	Understands that computers have no intelligence and that computers can do nothing unless a program is executed. Recognises that all software executed on digital devices is programmed.	Obtains content from the world wide web using a web browser. Understands the importance of communicating safely and respectfully online, and the need for keeping personal information private. Knows what to do when concerned about content or being contacted.	Uses software under the control of the teacher to create, store and edit digital content using appropriate file and folder names. Understands that people interact with computers. Shares their use of technology in school. Knows common uses of information technology beyond the classroom. Talks about their work and makes changes to improve it.
Can work with text, images and sound. Can work with text, images and sound. Can work with text, images and sound.	Recognises that a range of digital devices can be considered a computer. Recognises and can use a range of input and output devices. Understands how programs specify the function of a general purpose computer.	Navigates the web and can carry out simple web searches to collect digital content. Demonstrates use of computers safely and responsibly, knowing a range of ways to report unacceptable content and contact when online.	Uses technology with increasing independence to purposefully organise digital content. Shows an awareness for the quality of digital content collected. Uses a variety of software to manipulate and present digital content: data and information. Shares their experiences of technology in school and beyond the classroom. Talks about their work and makes improvements to solutions based on feedback received.
Can collect and analyse data. Can collect and analyse data. Can collect and analyse data.	Knows that computers collect data from various input devices, including sensors and application software. Understands the difference between hardware and application software, and their roles within a computer system.	Understands the difference between the internet and internet service e.g. world wide web. Shows an awareness of, and can use a range of internet services e.g. VOIP. Recognises what is acceptable and unacceptable behaviour when using technologies and online services.	Collects, organises and presents data and information in digital content. Creates digital content to achieve a given goal through combining software packages and internet services to communicate with a wider audience e.g. blogging. Makes appropriate improvements to solutions based on feedback received, and can comment on the success of the solution.
Can evaluate the quality of digital content. Can evaluate the quality of digital content. Can evaluate the quality of digital content.	Understands why and when computers are used. Understands the main functions of the operating system. Knows the difference between physical, wireless and mobile networks.	Understands how to effectively use search engines, and knows how search results are selected, including that search engines use 'web crawler programs'. Selects, combines and uses internet services. Demonstrates responsible use of technologies and online services, and knows a range of ways to report concerns.	Makes judgements about digital content when evaluating and repurposing it for a given audience. Recognises the audience when designing and creating digital content. Understands the potential of information technology for collaboration when computers are networked. Uses criteria to evaluate the quality of solutions, can identify improvements making some refinements to the solution, and future solutions.
Can design and create digital content. Can design and create digital content. Can design and create digital content.	Recognises and understands the function of the main internal parts of basic computer architecture. Understands the concepts behind the fetch-execute cycle. Knows that there is a range of operating systems and application software for the same hardware.	Understands how search engines rank search results. Understands how to construct static web pages using HTML and CSS. Understands data transmission between digital computers over networks, including the internet i.e. IP addresses and packet switching.	Evaluates the appropriateness of digital devices, internet services and application software to achieve given goals. Recognises ethical issues surrounding the application of information technology beyond school. Designs criteria to critically evaluate the quality of solutions, uses the criteria to identify improvements and can make appropriate refinements to the solution.
Can justify the choice of digital devices and services. Can justify the choice of digital devices and services. Can justify the choice of digital devices and services.	Understands the von Neumann architecture in relation to the fetch-execute cycle, including how data is stored in memory. Understands the basic function and operation of location addressable memory.	Knows the names of hardware e.g. hubs, routers, switches, and the names of protocols e.g. SMTP, iMAP, POP, FTP, TCP/IP, associated with networking computer systems. Uses technologies and online services securely, and knows how to identify and report inappropriate conduct.	Justifies the choice of and independently combines and uses multiple digital devices, internet services and application software to achieve given goals. Evaluates the trustworthiness of digital content and considers the usability of visual design features when designing and creating digital artefacts for a known audience. Identifies and explains how the use of technology can impact on society. Designs criteria for users to evaluate the quality of solutions, uses the feedback from the users to identify improvements and can make appropriate refinements to the solution.
Can undertake creative projects that collect, analyse, and evaluate data. Can undertake creative projects that collect, analyse, and evaluate data. Can undertake creative projects that collect, analyse, and evaluate data.	Knows that processors have instruction sets and that these relate to low-level instructions carried out by a computer.	Knows the purpose of the hardware and protocols associated with networking computer systems. Understands the client-server model including how dynamic web pages use server-side scripting and that web servers process and store data entered by users. Recognises that persistence of data on the internet requires careful protection of online identity and privacy.	Undertakes creative projects that collect, analyse, and evaluate data to meet the needs of a known user group. Effectively designs and creates digital artefacts for a wider or remote audience. Considers the properties of media when importing them into digital artefacts. Documents user feedback, the improvements identified and the refinements made to the solution. Explains and justifies how the use of technology impacts on society, from the perspective of social, economical, political, legal, ethical and moral issues.
Can understand the ethical issues surrounding the application of information technology. Can understand the ethical issues surrounding the application of information technology. Can understand the ethical issues surrounding the application of information technology.	Has practical experience of a small (hypothetical) low level programming language. Understands and can explain Moore's Law. Understands and can explain multitasking by computers.	Understands the hardware associated with networking computer systems, including WANs and LANs, understands their purpose and how they work, including MAC addresses.	Understands the ethical issues surrounding the application of information technology, and the existence of legal frameworks governing its use e.g. Data Protection Act, Computer Misuse Act, Copyright etc.

could be taught to achieve each Progression Pathway statement and National Curriculum point of study.

demics from the wider CAS community.

for Key Stage 3. Visit www.hoddereducation.co.uk/compute-it for more information.



HOW DO YOU START **GETTING TO GRIPS** WITH THE COMPUTING REQUIREMENTS?

Many primary teachers will be spending the summer term preparing for September. Rachel Iles shares her experiences of planning for the new computing curriculum. Rachel is Learning Technology Manager at Junior School Leidschenveen (The British School in the Netherlands).

DON'T REINVENT THE WHEEL: MAKE IT **STRONGER** INSTEAD

Remember, the new curriculum is not only programming. A broad curriculum nurtures students' skills, confidence and curiosity to explore different technologies. In KS2, especially since iPads are 1:1 in Y6, with a range of media at their fingertips, students are now better able to consider 'Which tool is best for the task?'. The ongoing challenge is not the 'what' but the 'how' of 'Literacy and Citizenship'. Appropriate digital use, research, sharing and identity balanced alongside organizing, storing, manipulating and retrieving. Key questions: Why use a keyword search? Why do we refer to the top 2 websites on a search page? What suffix do we notice on web addresses?

Through an ongoing and lively discussion (staff meetings, parent and student workshops), we have been working towards greater behaviour awareness. I feel a definite shift from technology to 'Citizenship' – doing the right thing and being aware of your digital behavior and identity. Key questions: What makes a good photograph? Why review them? How do we delete images that we will not use? Etiquette – Do we need to check/get permission before taking a photograph? Can we delete any images on a device, or only those we have taken ourselves? What do you do if someone shares a photo without you feeling ok about it? What do we do if we encounter something that we are unsure of, concerned about or makes us nervous?

We are not reinventing the wheel, but making it stronger by adding depth and breadth. The CAS network is a valuable professional forum through which you are able to get some fast advice, share information and be pointed in the direction of reliable, tested resources.

Whilst trying to get to grips with Computing and making links with what we already have in place, it became obvious we would not need a global redesign, but some shifting to integrate more specific vocabulary and programming in upper KS2. Our students currently explore coding with Scratch in Y4, BeeBot control in KS1, Daisy Dinosaur in FS, 2D>3D Design and 3D Game Club and I lead a team of enthusiastic Digital Leaders in a variety of roles throughout the school.

So – where do you start? I ran Spring INSET on how to further embed existing planning and during my workshops with each key stage team, a couple of comments helped me to think about the shift as a 'translation' issue. The biggest question that resonates from teachers is 'HOW on earth do I fit it in?'. Changing the mindset of embedded learning, with technology skills being integrated into cross curricula activities is key. How? Make it exciting, but manageable, and start with bite-size phases.

I'm introducing Daisy the Dinosaur in the FS2 as an alternative to BeeBots. Initial program commands are practised through role play, then using small group work to try it out, leading, where appropriate to students extending skills through the 'challenge mode'. Children love technical words and when modelled well and often, adopt them easily. The challenge doesn't come from the children but the adults who react to new terminology or the perception of 'programming' as not being directly appropriate for young learners — as did I initially, despite being technology positive. Is it the language that is scary? – I'd say yes, so small steps are key to getting people on side. I'm adjusting objectives and vocabulary as we move forward. Therefore the instructions "Control simple games on-screen using a range of program commands. Begin to debug programs and adapt commands to create an alternative behaviour" would be translated into: "Did your dinosaur do what you wanted it to do? Yes? - Can you make it do something different? No? - Can you see why, change it and make it do what you wanted?".

Last year Y4 students picked up Scratch very quickly. Fizzing and asking peers 'How did you do that?'; 'Can you show me?'; I decided to let them move at their own pace. It provided a fun way to create FairTrade slogans, bringing animation, sound, movement, backgrounds and sprites together. Can FairTrade slogans be translated as: "Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts."? Most definitely! Next year I'll spotlight transition to new technical vocabulary and build the terminology into self-assessments. Challenging concepts: 'sequence, selection, repetition and variables' lend themselves as well to the creation of a music score on Apple's Garage Band – where writers 'build in change' for alternative pieces of music to be applied, to designing a planet in Kodu or a control sequence for Mindstorms. I feel excited about changes ahead, and 'espresso +' and an online course or two have become regular Sunday friends!

UK KODU KUP WINNERS ATTEND INSPIRING WE DAY IN CALIFORNIA

Last July, three girls from Afon Tâf High School in south Wales won the inaugural Microsoft Kodu Kup. Head of ICT, Rich Thomas, explains how their success led to a trip to California, and helped inspire thousands of students through We Day.

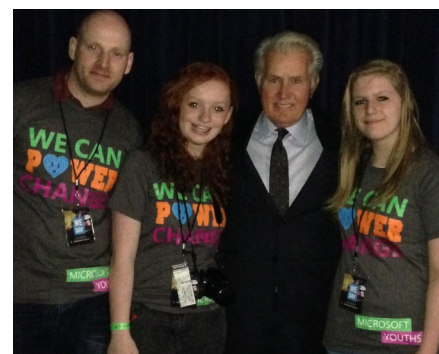
The first Kodu Kup competition challenged participants to code and create a game using the visual programming language Kodu. The competition saw the young women – Kayleigh Bennett, Holly Bridges and Shauna Coates, also known as Artemis Games – beat stiff competition from all areas of the UK with their game, “The Dark Side of Mars”, which explored the complexities of the human psyche and emotional wellbeing. From 154 entries, through to 11 finalists Artemis Games won. In recognition of their achievement, they each received an Xbox 360 console with Kinect and a trip to Lionhead Studios, an award-winning British game developer that produces the popular game Fable.

The premise of ‘The Dark Side of Mars’, involves a failed rescue mission to Mars. After a crash landing, the captain realises he is the lone survivor, and there is something chasing him. As he overcomes various challenges, he encounters the souls of his crew members, realising that the monster chasing him is ‘Regret’. His mission is to come to terms with his responsibility for their deaths by rescuing the souls. The girls were not just required to design and create this complex game. They also had to present and promote it to an audience,

including a judging panel comprising some of the most revered game development experts.

In March, 2014, the team joined co-sponsors Microsoft as VIP guests at the Oracle Arena in Oakland, California. As young innovators, they helped raise the profile of women in the computer games industry and served as a great example of what youth can accomplish if they follow their passions and dreams. We Day is an initiative of Free The Children, an international non-profit organization that brings youth and educators together with actors, musicians and other speakers through an inspirational event and yearlong school based program called We Act. Microsoft’s support of We Day and the We Act movement aligns with the company’s YouthSpark initiative to create opportunities for youth that empower and inspire them to change their world.

This year We Day returned to Seattle, Washington, and made its debut in California and London, England. In California the girls shared the same stage as celebrities and singers including Selena Gomez, Natalie Portman and Orlando Bloom. Learn more about their experience at www.microsoft.com/weday/.



THE KODU KUP

The Kodu Kup is running again, following the success of last year’s competition, but with a few changes. Students can now enter using new software, Project Spark, developed by the same team as Kodu Game Lab. It is used in the same way with “When” and “Do” instructions but gives more scope with the design and coding, including a range of variables, loops and Boolean logic.

The Kodu Kup is open to anyone from a UK school aged between seven and fourteen. Children must be entered as a team of three, forming a mini “game studio”. The closing date is Friday 30th May. The top three teams in each category will receive an invitation to present their games to a panel of judges from the games industry at Microsoft Headquarters in Reading. At the end of the day the winners of each category will each receive a Surface RT tablet and the overall winning team will take home the Kodu Kup! Last years’ video gives additional information: <http://youtu.be/XOHhCJddeZY>

Nicky Cooper

A SHORT INTRODUCTION TO THE INTERNET AND THE WEB

How do our browsers and the web actually work? How has the Web evolved into what we know and love today? And what do we need to know to navigate it safely and efficiently? “20 Things I Learned About Browsers and the Web” is a very short, child friendly e-book, produced by the Google Chrome team. It looks first at the Internet, the very backbone that allows the web to exist, then how the web is used today, through cloud computing and web apps. It introduces the building blocks of web pages, such as HTML and JavaScript, and reviews how their evolution has changed the sites you visit. It also takes a look at browsing the web more safely and securely. An ideal quick introduction for time pressed primary teachers looking at this for the first time: www.20thingsilearned.com





STRENGTHENING MATHS, SCIENCE AND COMPUTING THROUGH MATHEMATICA

Computers are used to solve a huge number of different problems in fields from meteorology to finance, and the application of programming to these types of problems can be a powerful motivator for students. Nick James, Director of E-Learning at Ashford School in Kent praises a powerful tool for doing just that.

Wolfram's Mathematica software offers access to both curated sets of real world data and powerful built in algorithms to manipulate this data. It is also available for free on the Raspberry Pi, and is included in the newest versions of the Raspbian operating system. Historically, Mathematica has been most widely used for applications in Mathematical and Scientific Computing. It shines in this role, and therefore in the Mathematics classroom, because of its ability to symbolically manipulate algebraic objects. This enables Mathematica to function as an extremely powerful graphical calculator, with real time manipulation of various parameters through its built in Manipulate function. Many schools have adopted Mathematica for this reason alone, but that only scratches the surface of its capabilities.

Other features make Mathematica a powerful tool for teaching programming and computational thinking, both within and outside traditional Computing lessons. It provides code highlighting so teachers and students can see quickly where possible errors are in their code. It also has a predictive interface suggesting func-

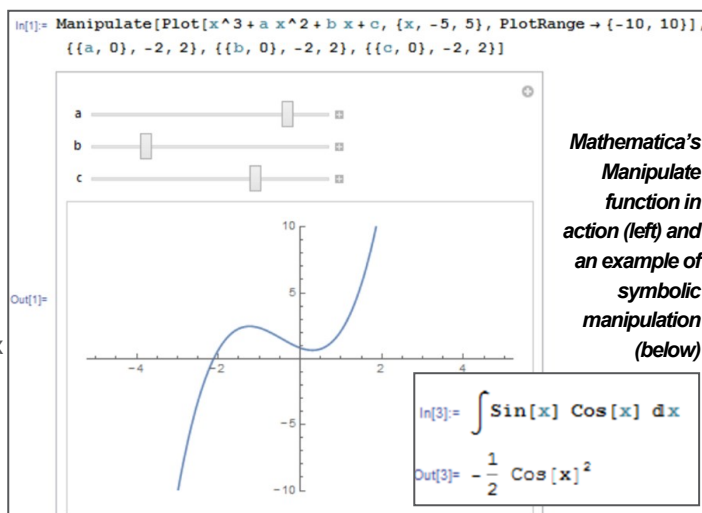
tions a user might like to use. For instance, if a calculation has resulted in a function or a matrix, it will offer to plot that function or produce a diagram of the matrix to allow further analysis.

Mathematica provides immediate feedback and graphics. In just a few lines of code, and without the need to import any libraries, pupils can construct 3 dimensional graphics and animations, see the results, then alter their code to improve it. This all happens in the same window.

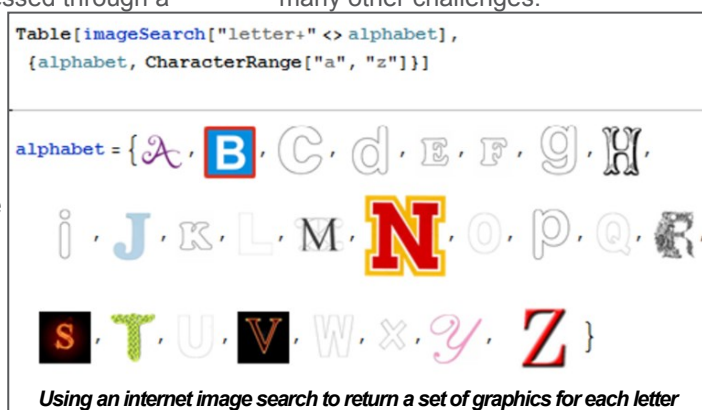
Mathematica also comes with access to a huge number of built-in, curated data sets, all accessed through a common interface. Pupils can learn to manipulate these with a few simple commands to produce tables of statistics on countries, comparisons of elements and compounds, and comprehensive weather reports from any time or place, to give just a few examples. The documentation is comprehensive, easy to understand, comes with many code examples, and is available inside the coding window.

All of these factors combine to create a tool which allows motivated students to answer genuinely interesting ques-

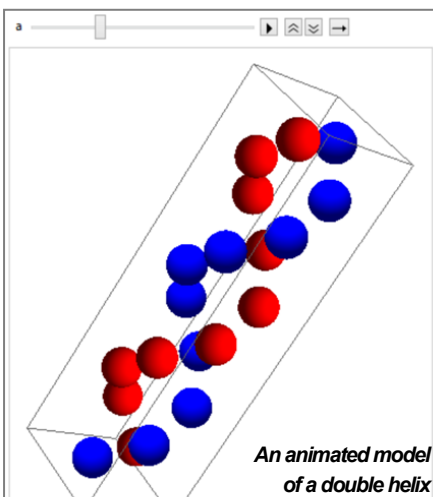
tions about the real world quickly and efficiently. My pupils, in mathematics lessons, have produced animated models of a DNA helix, calculated the distances between cities, and plotted that line on a map, written code that downloads results from a Google image search to produce a program which creates ransom-style notes from any entered text, along with many other challenges.



Mathematica's Manipulate function in action (left) and an example of symbolic manipulation (below)



Using an internet image search to return a set of graphics for each letter



An animated model of a double helix

USING **TANGIBLE MODELS** TO HELP CHILDREN'S UNDERSTANDING

Many computing concepts can be made concrete through the use of physical models pupils can play with. Paul Revell, Head of ICT at The Lakes School in Cumbria shares a couple of ideas.

When children start to use number, we would expect to provide them with counters of some form. They need to group objects on a table, break them up and re-form them. I did this myself as a 5yr old in the 60's with cowrie shells which were kept in an old tobacco tin in my tidy box. It's probably not seen as environmentally acceptable to harvest cowrie shells for this purpose these days and even less so to use your old baccy tins with the infants. However, the concept of providing a physical model of an abstract concept holds good. Those who went through traditional teacher training will be very familiar with Piaget's stages of cognitive development. The problem seems to be that some of my Y10s are still somewhere between the sensorimotor and pre-operational phases when it comes to VB and assembly language. With this in mind, I have got physical with my Computing classes. My simplest model is a length of hardboard to which I have glued 10 plastic cups. The cups are labelled 0 - 9 and they hold card or paper strips to represent data. It did not take long to make, but can be very handy when students get in a muddle with loops, substrings and arrays.



A bigger project has been a physical representation of The Little Man Computer, which forms part of the OCR GCSE controlled assessment. This started after a few frustrating lessons and lots of scribbled sketches on A3 paper as students wrestled with the concepts. I wanted to be able to physically move data around from RAM to processor and to have a bigger visual-

isation of how machine code instructions and addresses were handled. My version is 'hard wired' to perform a multiplication using 2 inputs no greater than 5. It comes with lots of cut out numbers which can be moved around. Other features include the 'crib corner' where the control unit can look up the opcodes, a paper clip program counter, a yoghurt pot cache and a few bits of a broken laptop to link to real hardware.

As I had hoped, using the model to 'compile' and 'run' the multiplication program did provide a tipping point in understanding for some students. What surprised me more was how much I ended up using my oversized LMC. It turned out to be useful in more contexts than I had realised, at both GCSE and A level. Generations of languages, binary numbers, computer architecture and the fetch execute cycle are obvious examples. You can find a high resolution version on the CAS Community at bit.ly/1k1j76D.

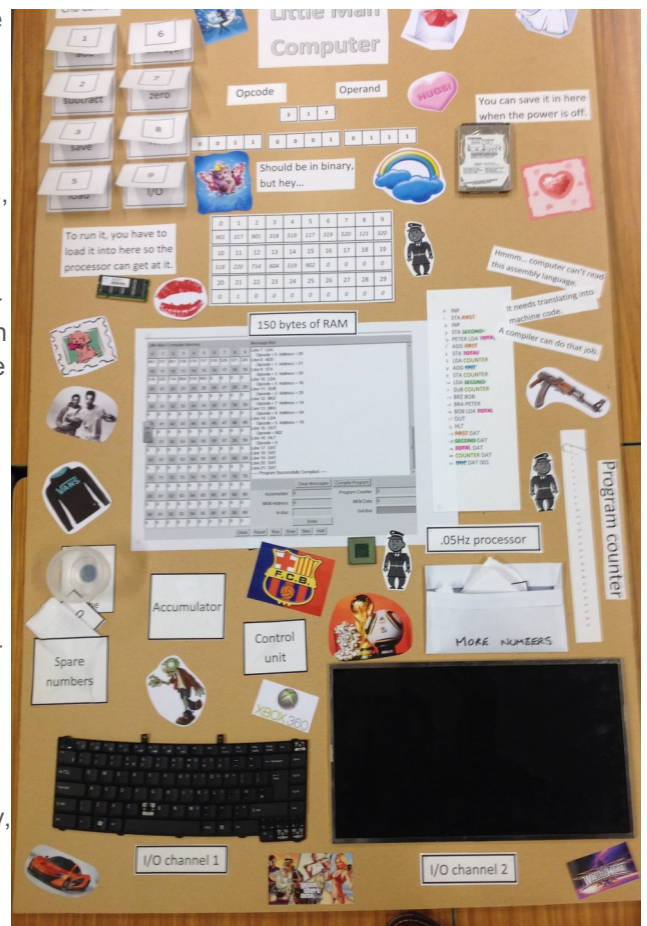
I have not taught much Computing at KS3 but I can see that physical models might well come in very handy. If the parents' stocks of empty plastic tubs and shoe boxes have not been exhausted by the Y7 castles project in History, I may well be needing them! Do other readers have ideas to share?



SIX PROJECTS RECEIVE **GOOGLE RISE AWARDS**

Google offers the [RISE Awards](#) to support organizations around the world working hard to inspire the next generation of Computer Scientists - especially those that reach girls, underrepresented minorities, and students who face socio-economic barriers. Our RISE partners are changemakers: they engage, educate, and excite students about computing through extracurricular outreach.

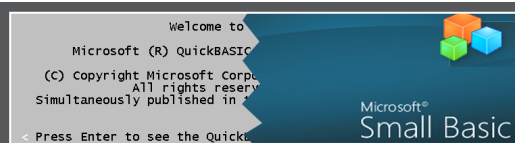
Six of the 42 organisations Google has chosen to fund and partner with globally are in the UK this year. Some will be familiar initiatives to CAS members. If not, check them out with the links below. It's very much a testament to how much great work is taking place in the field of Computer Science Education: [Teen Tech](#), [Generating Genius](#), [CAS #Include](#), [Ready Salted Code](#), [Code Club](#) and [Technology Will Save Us](#). Congratulations to all involved.





CELEBRATING 50 YEARS OF BEGINNERS ALL PURPOSE SYMBOLIC INSTRUCTION CODE

May 1st 1964 saw the release of the first ever version of the BASIC programming language. On the fiftieth anniversary CAS Master Teacher John Hughes, who teaches at Bishop Stopford School in Kettering looks back in admiration at the impact the language has had.



1964: Top of the Pops; Jackie magazine; BBC 2; and Donald Campbell, world speed record on water. Not so well-known was that I started secondary school in Leicester, and “across the pond” a small team of educationalists released the world’s first computer language aimed at teachers of programming, BASIC! For a moment just focus on the last and first words of the acronym: *Code* and *Beginner*. This was real code: we were in full control of the computer (for all the adventure and frustration that this would mean) and it was a language purpose-built for us - the fledgling programmers - the ones who were till then overlooked in the high-flying circles of either commerce or academia.

At a time when the only other choices were between Assembly language, or FORTRAN and COBOL, BASIC was the proverbial breath of fresh air. Why?

- few commands and simple syntax
- interpreted - no waiting for desperately slow compilation in those days - and so ...
- immediate feedback - why else would you program for pleasure?!
- “All-purpose” - okay, so not a universal language, but it freed the user from both rarefied mathematical problem-solving and hard-nosed commercial transactions, neither of which were ‘beginner’ activities.

Recreational programming was around even then, from the dubious 5 feet long pictures printed on teleprinter stationery to Robot Escape to Bulls & Cows (a forerunner of the hand-held Mastermind game), to Star Trek and the early multi-user games. And remember this was on machines that filled rooms, rooms that you weren’t allowed into, so in both a literal and figurative sense: remote programming.

Fast forward a decade or so to the 1970’s and 80’s and the home computer boom. Fast ROM - immediate start-up. Imagine! You press ‘on’ and it just starts - bliss! Simple (monochrome) graphics, then low-res colour, then hi-res colour, then layered graphics, ... leading to an endless supply of new games. Just buy the magazine and type it in, then wait for next week’s errata! Or for a *grown-up* challenge: take a game written for one machine and adapt it for your own. Ah, the black art of non-portability-hence-enforced-adaptation. BASIC had global appeal - the craze hit both sides of the Atlantic and made significant inroads to the lands that our cousins seem not to have heard of: Europe. Admittedly we were more aware of crazes in our local computer club or UK magazine, but the ferment of creativity was intoxicating and new, NEW, N-E-W.

So what were the killer features? A minimal core of must-have commands that included print, tab, input, for..next, if..then, read..data, user-defined single-line function, gosub..return, and yes, line numbers. But it was really that the ‘hobby’ package (the machine, its Operating System and BASIC - the only application available) was so intimately entwined it gave them the edge when it came to programmers’ creativity:

- Hardware screen-scrolling automatically gave an animated screen.
- *peeks & pokes* so you could interact **directly** with memory.
- *get & inkey* to allow single-character control by the user with both waiting and skipping possible
- computed *gosubs* and *gotos* to allow sophisticated program flow
- *tron..troff* to aid program-debugging
- Variable names permitted to be several characters long but only the first two were significant, leading to some very interesting bugs when they invisibly collided: *mygirth* was inadvertently equal to *myheight*, daily becoming an actual reality I regret to say!

But the very limitations, by today’s standards, of what was on offer were what gave these pioneering bits of kit their appeal: manageable output, (typical screen resolution: 30 lines of 40 character plus 300 by 200 pixels - the Haiku of coding, perhaps), double-height text for emphasis, tiny memories, logically divided (with the notable exception of the first Apple and TRS80 machines) so fully comprehensible by the amateur hobbyist. To my mind BASIC’s contribution to Computer Science’s younger enthusiasts is incalculable. Where would the RML & BBC machines have been, where indeed would Logo and Scratch have been without the programmers who were often initially motivated by their dabbling in that first ‘universal’ language, BASIC? So I offer an alternative mnemonic: BASIC = *Brilliantly Accessible; Stunningly Inspired Creativity*. See Dartmouth College’s celebration of BASIC at 50 at <http://bit.ly/1hCxq1p>

BASIC AT 50

APPROACHES TO PROGRAMMING: ARE YOU LANGUAGE AGNOSTIC?

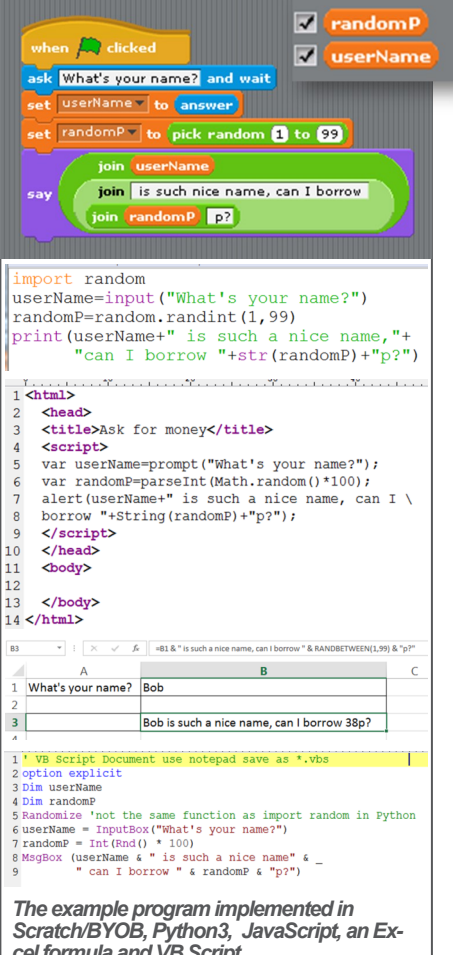
One way to make sure programming doesn't just become 'type in whatever is on the sheet' is through 'Language Agnostic' problem solving suggests Ilia Avroutine. In other words, teach a few languages and do the same problem in a number of them.

Computational thinking is not programming. Of course, programming shouldn't replace computational thinking, as they work together really well. Pressing "Run" is often the best way to validate good thinking. Using Language Agnostic problem solving, I would argue, it is almost impossible to do the same problem in multiple ways and not understand the Computational thinking behind it. For example, let's write a program that asks a user for their name, generates a random integer, compliments the user on their name and adds "can I borrow x p?" where x is the random number generated. This is usually a crowd pleaser and pupils can get creative with excuses and model little behaviour tricks involved in borrowing small amounts of money. It can be stated in pseudocode thus:

```
BEGIN BorrowRandomP
INPUT userName
SET P TO RANDOM 1 to 99
OUTPUT userName + "is such a nice name," + " can I borrow " + P + "p?"
HALT
```

Once we implement this program in one language (the language you are most comfortable with), we could teach pupils to do the same thing in other languages and write a little evaluation of which one was (a) most difficult (b) most fun (c) fastest (d) efficient (e) their overall favourite. Another benefit of this technique is that it mimics the industry practice – first the problem is set, flow-charted/pseudocoded and then the best language is decided.

For starters, pupils can play "spot 5 differences"; for scaffolding they can be given all bits of code fully worked out to code in, then given another task which recycles the same elements but this time only one language's code will be provided to them. Or possibly withhold some bits of code and let them figure it out. Plenty of extensions here! This approach get pupils asking a lot of questions, and as tiring as it is when pupils ask lots of questions, it's a good thing, isn't it? It also supports the conversation about the syntax of languages which is on the syllabus of GCSE Computing and upwards.



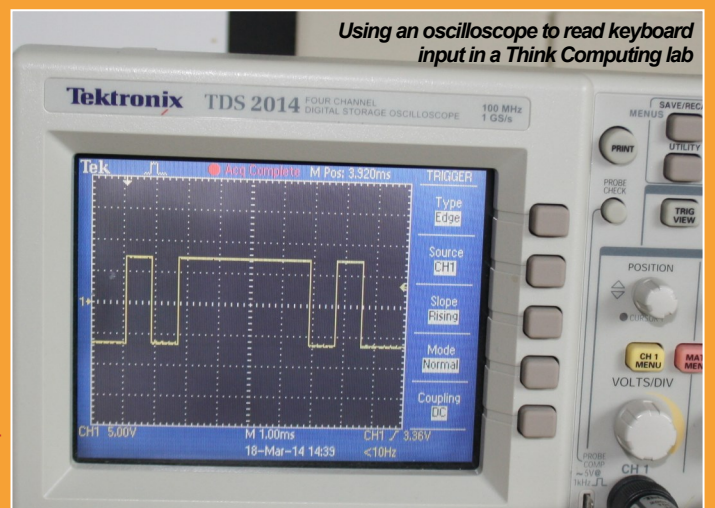
The image shows three different implementations of the 'BorrowRandomP' program. At the top is a Scratch script with blocks for 'when clicked', 'ask What's your name? and wait', 'set userName to answer', 'set randomP to pick random 1 to 99', 'join userName', and 'say join is such nice name, can I borrow join randomP p?'. Below this is a Python3 code snippet using 'import random', 'input()', 'random.randint()', and 'print()'. At the bottom is a VB Script code snippet using 'option explicit', 'Dim userName', 'Dim randomP', 'Randomize', 'InputBox()', 'Int()', and 'MsgBox()'. To the right of the Python3 code is an Excel spreadsheet showing the program's output: 'What's your name? Bob' and 'Bob is such a nice name, can I borrow 38p?'. Below the Excel spreadsheet is a caption: 'The example program implemented in Scratch/BYOB, Python3, JavaScript, an Excel formula and VB Script'.

Computing professionals I know rarely work in one language, often switching depending on the task or team. The languages shown are just examples. Can you get your pupils to use these, and others, to create their Rosetta Stone of languages? To achieve that, your pupils will have to solve the problem of constructing a multi-sided "stone" out of paper, the sides of which will bear the timeless inscriptions of Code.

TEACHERS JOIN FORCES WITH THE UNIVERSITY OF LANCASTER TO THINK COMPUTING

In March, around fifty secondary school teachers attended a day hosted by the University of Lancaster, School of Computing and Communications to Think Computing! The day included a keynote introduction and three hands on workshops using the Raspberry Pi to demonstrate concepts ranging in scale from packets of data to distributed algorithms. Demonstrations of research projects and a tour of the University data centre rounded off a packed day. Attendees took home their Raspberry Pi and packs of classroom resources as a final gift.

The University hopes this will mark the start of a closer collaboration with teachers in the locality, through the developing Network of Excellence, as they gear up to introducing the new curriculum.



Young Rewired State organises an annual Festival of Code (FoC), a week long event in the Summer holidays. Siobhan Ramsey, director of Sandbox Education (a professional development & web agency) and Young Rewired State Mentor explains what it is all about.

This year the Festival of Code runs from 28 July to 3 August. Students who are keen to use or develop their programming skills, can attend regional 'pop-up' centres across the UK, to design and prototype computing projects. They are free to work alone or collaborate, to build any type of game, app, web site or hardware project so long as they integrate at least one set of open data. They can choose to opt in or out of the YRS competition which takes place over the weekend.

On Friday, each year, everyone converges from the regions, for a long weekend, at the central Festival space, hosted this year by Plymouth University. Over the weekend the competition progresses through a series of heats while entries are judged by show and tell presentations delivered to peers and judges. There are talks, free pizza and a seemingly endless supply of ice-cream. This year a chiptune concert features the renowned @Pixelh8 (<http://www.pixelh8.co.uk>) performing music reminiscent of vintage computer games.

The atmosphere is industrious, informal and friendly, while tension mounts as the heats draw to a close and the winners are announced on Sunday. In 2013 there were 28 prize winning entries e.g. A Virus Spread Simulator and The Picycle, a service to provide navigational and other information to cyclists using a series of handlebar-mounted LEDs and a Raspberry Pi. All can be viewed at bit.ly/1dxEHL

The Festival of Code is free for 8-18 year olds and all work is done by volunteers. It strives to be inclusive (there is a hardship fund for help with travel costs) and can be an inspirational experience for all. Learn more at <https://youngrewiredstate.org>.

SECONDARY SUPPORT FOR PRIMARY **ROBOTICS DAYS**

A current challenge faced by many, is the adoption of technology new to the school and staff. Lyndsay Hope, from Monmouth School reviews a successful collaboration with a local primary school.

One means of addressing familiarity with hardware is through projects run in partnership between Secondary and Primary schools. For example, Year 5 pupils at Osbaston Church In Wales School worked on a Mindstorms robots activity day, supported by a small team of students and staff from Monmouth School. The event comprised a series of project modules, each relating to an aspect of robotics but also exploring other areas of problem solving. Each module had a particular focus from a Robotics Intro, through Building, Control, Sensing and Programming and culminating in a race track competition where pupils could test their builds. Pupils worked in small teams, each equipped with basic bot parts, plus extension components and guides.

Pupils began with an introduction to robotics, exploring why they can be helpful and what some of their limitations may be; then in their teams built a basic tri-bot. In the Control Task children discovered how to control the bots using mobile devices and a neat little Android Mindstorms app. They experimented with directional movement, then did a driving test where each child steered the bot through one drive, park, reverse and sprint loop. This gave lots of opportunity for exploring testing and development, understanding handling limitations and how control may be refined.



In the Sensor Task pupils added an ultrasonic sensor to the bot to help it respond when, for example, it approached a wall, and discovered how sensors can be used to trigger a response; they added sounds to the bot to give an alert when the sensor is triggered. The Programming Task introduced on-brick programming and pupils extended their sensor use by programming the robots to reverse, turn then roll forward again when they sensed an obstacle. The group played break-out games, forming a circle then setting their robots running in the middle to see which robot could sense a break in the circle and escape first.

Finally the teams competed in a timed track race, each team steering their bot around a race track, working to control their robots around bends and make up speed on the straight, discovering the balance between speed and accuracy. The modules were interspersed with brief video clips of the Big Dog robot, and Will.I.Am and others encouraging children into STEM careers. The model worked well with senior students gaining much from supporting juniors, and juniors being able to use resources that may not otherwise be available to a primary school. We're now looking at putting together a similar Raspberry Pi event and enthusiasm from all involved is high.

SCHOOL/UNIVERSITY PILOT PROJECT: ADOPTABOTS MAKE CODING COOL

CAPTURING and keeping pupils' interest in learning to code are key drivers of a pilot project between Brunel University's Department of Computer Science and four London schools, now set for a nation-wide roll-out. Brunel's lead academic Dr Stasha Lauria explains.

We wanted to look beyond the immediate needs of the curriculum changes and provide an off-the-shelf, cost-effective, compelling experience for pupils across a range of ages and abilities that was not tied to a particular language and offered other pedagogic benefits beyond learning to code. The result was Adoptabot.

It's built on our experience with the finch robot system with our first year computer science students. Each of them gets issued with one on day one of their course. Finches have a 'beak' that can change colour, an accelerometer so the orientation is known, IR sensors to detect obstacles, a light sensor, a thermocouple to determine temperature and a buzzer. So they are pretty cool. With Adoptabot pupils have to develop their own scenario where as many of the finch's capabilities are programmed in as possible. In the pilot we had scenarios and sets that could not have been more wildly different – the only limit being a team's imagination.

Another beauty of Adoptabot is that pupils in year seven or eight can see almost immediate results with no prior coding knowledge yet the finch is challenging enough for our undergraduates to use in final-year projects!

In the pilot we had students from years seven to thirteen equally challenged, enthused and engaged. In these days of incredibly tight school budgets that's an important consideration. And from the teacher's point of view it means he or she doesn't have to manage and juggle a variety of different projects. With Adoptabot it's one almost infinitely adjustable size that can fit all. Each participating school was allocated undergraduate



computer science student mentors from Brunel. They too benefit. We saw our students gain in self-confidence and improve their communication skills. Next time they will become the managers of a small cohort of younger students who will be going into the schools. And eventually we hope to see Adoptabot being almost exclusively student-run.

From the school's perspective there was one unintended but universal benefit. Explains Andrew Doxsey, Key Stage 5 Co-ordinator at Hounslow's Heathland School. "We've all got pupils who can face issues because they are seen as geeky or nerdy by their peers," he said. "When they are part of Adoptabot those interests and skills that may have set them apart are suddenly seen as intensely valuable by the other members of the team."

Brunel aims to take the scheme nationwide with support available 24/7 via a wiki. Although I believe the school/undergraduate mentor model could easily be replicated with clear benefits for both sides.

GOOGLE FUNDING FOR TRAINING IN WALES

Welsh CAS Hubs and Universities have been awarded \$25,000 funding from the Google 'Computer Science for High School' (CS4HS) initiative to deliver essential teacher training across Wales this summer. The project, led by Glyndwr University in Wrexham, and including Bangor, Cardiff Metropolitan and Swansea Metropolitan Trinity St David Universities, will allow the four centres to offer a residential weekend in July where teachers can get hands-on experience of programming in Python and working with the 'Little Man Computer' (LMC) simulator. The programme will be very practical in nature. Up to 15 teachers can be supported at each location with the full cost of training, accommodation, food, drink and reasonable travel met by the Google grant.

Glyndwr's Professor Vic Grout, North East Wales Hub Leader, said, "We're delighted to have secured this money for Welsh CS teaching and we're very grateful for the support given to us by CAS in submitting the proposal. This should be a great opportunity to offer some practical support across Wales and it's bound to be an enjoyable weekend".

The weekend's activities will begin at 4:30pm on Friday 4th July in Bangor, Cardiff, Swansea and Wrexham and follow a busy schedule, through to 4:30pm on Sunday 6th July. A fun part of the programme will be a live hook-up between the four locations. A games-based challenge, with both hardware and software content, will be set on Friday for groups to share experiences on Sunday. Participants should expect to depart Sunday exhausted but fulfilled! Places are limited and may be taken quickly so, to provisionally book a place, contact Vic Grout on v.grout@glyndwr.ac.uk

FIRST CAS NORTHERN IRELAND CONFERENCE TAKES SHAPE

Monday June 23rd will see the first CAS (NI) conference and the first CAS Conference outside the UK mainland. The theme will be 'Sharing good practice'. This exciting opportunity will allow all CAS(NI) colleagues to come together, exchange ideas and, most importantly, allow new colleagues from education and industry to join us and become future members. The opening keynote session, 'Insights From EdTech – A vision for Northern Ireland' will be delivered by the 3 Northern Ireland winners of the EdTech Award 2013; Gareth McAleese (GoBeserk), Roisin Crawford (STEM Aware) and Kerri McCusker (PhD Researcher, University of Ulster). They will share their insights from an intensive two week US exchange program which exposed them to global companies (Microsoft, Google), innovative institutions (MIT, KIPP) and game based learning companies (Emerson Game Lab, Bungie).



The workshops will have something to offer every attendee whether you are working at 'A'-level, or at KS2, whether you are in Education or in Industry. I am delighted at the response already expressed by educators in Northern Ireland about the conference with a real buzz starting to rise.

The cost will be £35 and application will be through the CAS(UK) website, events section. Northern Ireland is renowned for its hospitality and we would welcome colleagues from other parts of the CAS community. Anyone willing to offer a workshop session or simply wanting to come to Northern Ireland to hear what we are doing will be very welcome. For those of you who have never visited Northern Ireland, we have deliberately chosen Monday for our conference to afford people the opportunity to spend the weekend sightseeing before joining us at Stranmillis University College (above) for the event.

Irene Bell

TECHNOCAMPS RESPOND TO GROWING DEMAND FOR GCSE

A free intensive programme giving teachers help to deliver GCSE Computing across Wales is underway. Professor Faron Moller, Director of the Technocamps Programme at Swansea University explains the project.

Getting to grips with Python and Greenfoot with Technoteach at Swansea University



Technoteach, which is funded by the Welsh Government through the National Science Academy and run by the Technocamps programme at Swansea University, provides CPD in Computer Science to primary and secondary teachers. The sessions run for six weeks, three of those hands-on programming and the final three based around pedagogy. With 20 primary teachers through the programme before Christmas, the 2nd cohort is now underway with some 30 secondary teachers from across South Wales. Secondary sessions focus on using Python and Greenfoot as part of the new GCSE Computing. Kate Stephens, Head of ICT at Bishop Gore Comprehensive School, who recently started the programme, said "Technoteach provides a fantastic opportunity for ICT teachers to re-skill, for the changing curriculum of ICT/Computer Science, currently being rewritten in all syllabuses in Wales and indeed in England. As Head of ICT in Bishop Gore School, I have not dabbled with much programming since my degree over 12 years ago and after only one session of Technoteach I now feel confident to teach the basics of Python through lunchtime Technoclubs. I have already written some complex challenges which will be used to enthuse and inspire our KS3 pupils into choosing Computer Science at KS4!"

The demand for delivering GCSE Computing in schools across Wales is growing and the Technoteach project provides quality training for teachers, equipping them with the necessary knowledge to teach young people the fundamental skills of computing, required for the future digital economy. Sarah Thomas, Head of IT at Ysgol Gyfun Emlyn, said, "The school is always looking at courses which would interest pupils and decided this year to offer GCSE computer science as well as ICT. Staff in the department have needed a refresher training course on programming in particular. Members of the ICT department have attended other Technoteach events too. CPD events like this are very important otherwise we would not be able to offer the courses to our pupils." Further sessions will be run later this year, so if any primary or secondary teachers are interested there will be plenty of opportunity to sign up. You will need to commit to attending the full course which is free of charge and we will also pay for supply cover up to £80 for the first session.

GIRLS TECHNOVATION CHALLENGE SUCCESS LEADS TO DREAM U.S. TRIP

Technovation is a technology entrepreneurship competition for young women. Teams work together to develop mobile apps, then pitch their 'startup' businesses to investors. Theresa Russell, from Morecambe Community High School outlines how they got involved.



Last year, five girls from Morecambe Community High School – Elisabeth Y8, Erin Y9, Dione Y9, Nicole Y10 and Aimee Y10 – were selected to enter the Technovation Challenge by their ICT/Computing teachers, Ms Birkinshaw and myself, their brief being to design an App that would help to solve a community issue. They needed to work with a female mentor from industry. The girls were extremely fortunate that Silvia Spiva from Cisco, San Jose, California heard about their quest and contacted me via Twitter and offered her support. She also put the group in touch with Heidi Rhodes at Cisco, London. Cisco is a world-wide leader in networking that transforms how people connect, communicate and collaborate – Heidi immediately did all three of these by visiting the girls here at school for a day's business planning and to map out a strategy to take the project forward.

The girls decided to call themselves TechGirlsUK and name their app Re-Genz. Its purpose was to co-ordinate events between volunteers and those who need help with community projects. The journey the girls had embarked upon was now developing rapidly and they were invited by Heidi to visit Cisco in London to make a promotional video for their project. Whilst there they also held a live WebEx meeting with Silvia in San Jose, California, Kasia Nowaczyk and Petra Gnirk in Switzerland and Heidi's boss Chris Palermo from New York. With their video presentation and Business

Plan complete, they submitted their pitch and waited anxiously for the results. In the meantime the girls continued their fund raising events. They were really pleased to see donations and sponsorships flooding in from businesses, organisations and generous individuals. The group were very grateful to Ms Birkinshaw's son, Daniel, who organised a very successful Women of Tomorrow fund-raiser at the RBS/NatWest in Manchester.

The girls were invited to the Apps For Good event at Manchester University to talk about their journey. Unfortunately they did not get into the final of that competition but received an honourable mention, with one judge writing, "I love the problem that you have tried solving here. I also love the research you have put into what motivates people to do social good. Thumbs-up for trying to make the pitch humorous too!" Disappointment soon turned to delight, however, as Cisco generously invited and subsidised the group to visit their HQ in San Jose, California last May. The girls would like to extend their thanks to Heidi, Silvia and all those who helped make their wonderful journey possible. For full details of our trip to America please visit our blog at bit.ly/1icceAX. As I write, we have three teams of girls busily recording their final pitch videos for the 2nd May deadline. This has generated a lot of interest in GCSE Computing with Year 9 girls and is having a positive effect on numbers.

GIRLS INSPIRED BY TRIP TO SILICON VALLEY

2014 marked the third year of the annual computer science trip conducted by Townley Grammar School in Bexleyheath, Kent. Designed to encourage females into computing, it included visits to some large tech companies. Visiting Google and Facebook had to be the two landmark events of the trip. We were fortunate to receive inspiring and motivating talks from women working in the field as well as obtaining a stronger insight into the industry and the jobs available. The trip included visits to Microsoft, Intel and Netapp; three companies that work in very different areas of the industry. The pioneering work of Microsoft and Intel was highlighted and whilst touring Netapp our eyes were opened to the various careers in computer science.



In addition we visited certain high schools and universities. The professors at Stanford University underlined the importance of the industry and its ever evolving nature. The students saw the American curriculum, and have been encouraged to expand their skill set further outside of the classroom. Townley has established its own Coding and Robotics Club, in addition to participating in nationwide events such as BIMA Digital Day. We want to thank our Computer Science department for giving us the opportunity to participate. Our philosophy is very much like that of Robert Noyce: "Don't be encumbered by history. Go off and do something wonderful."

*Krupa Dodhia & Ella Hollis
(6th Form Computing & ICT Students)*

A PAUSE FOR THOUGHT

Big Data hit headlines some four years ago when Google published their tracking of a US flu outbreak in Nature magazine. Their identification was based on algorithmic identification of trending terms online and preceded that of the Centre for Disease Control by several days. This illustrated one facet of the exciting potential of big data. Governments, researchers and corporates alike are exploring how best to exploit the potential by extracting meaning from the data mass.

Supermarkets already use inference algorithms to tailor offers to individuals, with occasionally disturbing accuracy raising questions of privacy and intrusion. Predictive Policing, analysing data from diverse sources to anticipate future crime, further highlights questions of the use of big data: when does a potential crime justify intervention? Should possible victims be informed beforehand? Is the derived information accurate?

And what about health care and insurance companies? Services can be targeted to those most at risk making huge cost savings, but will we see elevated insurance premiums for those who expose their love of burgers via Twitter? Despite anonymization, individuals may be identified by the correlation of separate pieces of data, exposing information that is more private than that which they agreed to share. In some contexts the consequences could be tragic.

Meanwhile, Google is refining its algorithms following an inaccurate peak flu prediction, based on data distorted by a large outbreak early in the season. This field is fascinating but not yet infallible. See <http://www.google.org/flutrends/>

Lyndsay Hope

HOW COMPUTING CAN MAKE MEDICINE SAFER

The new cs4fn magazine (issue 17) explores how computer scientists save lives. Our hospitals are full of computers in disguise, keeping patients alive. To build machines that not only save lives but are safe you need to think about the big picture, not just the gadgets. Programming them is important but it is only a small part of what matters. Only when you know how they are really used can you find ways to make them even safer. We look at a research project called 'CHI+MED' where computer scientists are working with psychologists, social scientists and medical professionals to find and solve the problems. Other articles show how Formula 1 has led to safer operations, how taking a leaf from a gorilla watcher's book has helped one manufacturer get a market edge, and why programmers really are wizards! Order free copies from www.cs4fn.org/magazine/.



COMPUTING: THE NEXT GENERATION

Computing At School Sixth Annual Teacher Conference
University Of Birmingham
Saturday 21st June (reception 7pm, Friday evening)

The conference will have its usual mix of plenary sessions, over 30 different workshops, the opportunity to network and take home practical examples of lessons and other resources that you can use in your classroom.

"I learnt more and connected with more interesting people than on any course that my school has previously forked out hundreds of pounds for"

Cost: £35
To register visit bit.ly/1IGcpa7



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. **SWITCHED ON** 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: Ilia Avrouline, Irene Bell, Jo Brodie, Nicky Cooper, Tom Crick, Paul Curzon, Alison Daniel-Cutler, Claire Davenport, Andrew Daykin, Roger Davies, Peter Dickman, Mark Dorling, Emma Goto, Vic Grout, Taryn Hauritz, Lyndsay Hope, John Hughes, Rachel Iles, Nick James, Catriona Lambeth, Sam Lawrence, Stasha Lauria, Kevin McLaughlin, Faron Moller, Nevita Pandya, Siobhan Ramsey, Paul Revell, Matt Rogers, Theresa Russell, Sue Sentance, Adam Shelley, Andrew Shields, John Stout, Rich Thomas, Terry Watts and John Woollard.

www.computingatschool.org.uk

Computing At School are supported and endorsed by:



The Chartered Institute for IT
Enabling the information society

Microsoft®

Research

Google™

CPHC
The Council of Professors and Heads of Computing