COMPUTING AT SCHOOL
EDUCATE · ENGAGE · ENCOURAGE
Part of BCS, The Chartered Institute for IT

SWITCHED ON

COMPUTING AT SCHOOL NEWSLETTER SUMMER 2016

HELLO WORLD!

A new educational imperative is emerging. It's based not on economics, important though that is, but the premise that all children need exposure to Computer Science. We take it for granted that they will have a grounding in the natural sciences. It is a key entitlement, not because they will all become chemists, physicists or biologists but because everyone benefits from a basic understanding of how our world works. The days of superstition and magic are long gone. In ten years' time, the common claim that 'computers are magic' may seem as bizarre as a view today that the world is flat. Our world has been digitised. It needs citizens empowered by an understanding of how it works, not enslaved by technologies they regard as incomprehensible.

Teachers in the UK are pioneering new ways to teach Computing. We are not alone; this issue takes a special look at exciting developments taking shape across the world. Computer Science is fast gaining recognition as a discipline in itself and as an 'underpinning' subject for STEM. The message is clear; we are part of an historic educational sea change. No change is smooth. Navigating the way forward constantly presents new challenges, but CAS members can take heart (and pride) from their efforts so far. Your energy, openness and desire to share are moulding a new subject. Equally important, you are providing a model of professional development others wish to follow. The world is watching with interest.

INSIDE THIS ISSUE

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

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

WEST MIDLAND CRC FOCUSING ON ENGAGING HEAD TEACHERS

If you are reading this you are likely to be enthusiastic about the new Computing curriculum and the introduction of Computer Science. You can probably see the wider benefits for all your pupils but you may be struggling to get sufficient hours to teach it effectively. The approach of individual schools to the new curriculum changes differs widely. With increased academization, greater curriculum freedom, a variety of advice agencies, new performance measures and a challenging financial climate, schools are juggling a range of competing demands. Many have bitten the bullet and enthusiastically endorsed Computing. Others are more reticent for a whole variety of reasons. We want to reach out to these schools.

Do you feel that your Head and Senior Leaders don't really get Computer Science? Perhaps you feel unsupported in your school? If so, you are not alone and the CAS West Midlands Regional Centre in Birmingham is trying to do something about it. From mid afternoon on Thursday 28th June we are running an event specifically aimed at Headteachers, Senior Leaders and school governors, to showcase some great schools which have truly embraced the new curriculum. We will have speakers from CAS and Birmingham City University as well as Headteachers who have really got to grips with the change and are making it work in their schools. Please do let the CRC know if you want us to invite your Head, SLT and / or school governors. Our contact information and more details can be found on the CAS Community website, see events/3924.

Duncan Maidens

Despite competing pressures facing schools, Headteachers are aware of the changing needs of a modern economy. They will be attuned to the global movement now emerging that recognises the foundational place for Computer Science in a modern liberal education. Make sure you share this issue of SwitchedON with your school leaders. CAS will launch a national initiative to engage and advise school leaders, with suggestions of key steps to take to succeed in the near future. More in the next issue.

REGIONAL CENTRES MAKING IMMEDIATE IMPACT

The ten new CAS Regional Centres are fast establishing themselves as a key part of the mission to develop professional excellence. Zoe Ross provides a roundup of activities.

Offering expertise and support at a regional level, CAS Regional Centres (CRC's) are busy coordinating activity across their individual regions, holding special events, running training, and supporting Master Teachers, Hubs and Lead Schools. All regional centres have run, or are planning to run, micro:bit, Barefoot and Tenderfoot workshops, offering unique training opportunities for Computing teachers to access up-to-date resources and training. You can find information on all the regional centres, including contact details, on the CAS website: www.computingatschool.org.uk.

CAS North East has also been offering specialist help in physical computing together with A-Level Computer Science. It is offering schools in the North East workshops using Crumble bots in Primary phase and Arduino robots for Key Stages 3 to 5, with unplugged CS activities alongside.

CAS North West Lancaster has been out on the road supporting local Hub meetings, talking to teachers and promoting the regional centre. Its regional conference is on Wednesday 15th June and will have a physical computing theme, offering a variety of workshops for opportunities for Primary and Secondary teachers.

CAS North West Manchester ran a hugely successful conference in October and since then has been working hard to support teachers across the region. It has been running regular hands-on workshops and support groups for Master Teachers at Manchester and Edge Hill Universities. Sessions have included Python, App Inventor 2, Geocraft, SQL, CodeBug, 3D printing, Little Man Computer and A-Level support.

CAS Yorkshire and the Humber has been busy organising regional meetings and CPD opportunities galore for teachers in its region together with creating a curated set of resources for use by Hub leaders, Master Teachers and other CAS members. You can find more details at http://www.computingatschool.org.uk/crcs/yorkshire_and_the_humber

Lorraine Underwood (CAS North West Lancaster) at the Preston Hub

Carl Simmons from Edge Hill University (CAS North West Manchester) being 3D body scanned.

CAS East Midlands has been supporting teachers in lots of ways, including setting up an equipment library. Teachers in the region will soon be able to borrow equipment from the regional centre, including Raspberry Pi's; Crumble kits, bots and robotic vehicles, and micro:bits.

CAS West Midlands held a successful launch event for Hub Leaders, Master Teachers and Lead Schools in the region, including hands-on workshops for teachers. It has specific expertise around computer networking and cyber security and it is leading on the development of a set of resources to deliver many aspects of the new Curriculum. Once up and running, it will be freely available to schools.

CAS East has been involved in a range of activities across its region, including working directly with 15 secondary schools to deliver a Computer Science Enrichment Day, where Master Teachers and other experts led sessions for 180 Secondary students and their teachers on robotics, Raspberry Pi, app development and more.

CAS London's highlight was its first conference which was great fun! 170 delegates over 30 workshops; over 800 man hours of CPD! The best bit was seeing the most amazing collaborations and CPD, from Master Teachers and from other colleagues with varying levels of experience, working together with university lecturers and industry specialists to create a very special atmosphere and CPD-packed day.

CAS South West ran a successful regional conference in November. The keynote address by Dr Jo Twist is available on its YouTube channel: www.youtube.com/watch?v=8DMP0UkYMBY. Amongst its many activities it has been working on a successful partnership project between a Lead School, DAT (Plymouth University) and Microsoft, focusing on the widening participation of girls.

CAS South East has been out and about supporting hubs across the region and meeting CAS members. It has been running micro:bit and Tenderfoot training sessions throughout the area, and is planning a Micro-Computer Festival later on in the school year to celebrate pupils' success in using micro-computers to solve interesting problems. It also ran a very successful GCSE Computing day in December attended by over 60 teachers.

It's been a busy start but, through their activities, CAS CRCs are quickly getting known, establishing a dialogue with teachers and schools in their localities. We look forward to building on this exciting start and seeing what can be done in the next two years.

CAS TV

Thanks to the generosity of our friends at Google, CAS can announce the (re)launch of its YouTube channel, CAS TV. The new channel launched with Simon Peyton-Jones explaining algorithmic complexity (and giving his view on P vs NP), followed by Pete Kemp developing computational thinking through 3D animation, Alan O'Donohoe teaching Python, and Catherine Elliott of #include advising on students with SEN/D accessing the computing curriculum. Much more is planned for the weeks and months ahead; the subject knowledge challenges of Secondary Computing to link with CAS's Tenderfoot Computing project; practical strategies for teaching programming; and physical computing, to name a few. We hope to cover broader issues too, such as lesson observation, extracurricular projects and Primary - Secondary links. We'll also add to the archive of CAS conference presentations.

We're on the lookout for content and programme ideas: if you'd like us to come and film an interview about an interesting project, or (with the necessary permissions) film in your school, please do get in touch. We're also happy to add CAS members' own YouTube content to the CAS TV home page - so ping us a note if you'd like something included. We're keen to capture the CAS community spirit on the new channel: "There is no them - only us".

SUBSCRIBE NOW!

To keep up to date with all the new content as it arrives, simply click on the subscribe button on youtube.com/computingatschool.

HOW QUICKLY WE'VE ADAPTED LORNA ELKES' DIARY

Our Chromebooks are constantly in use across the school and the impact of access to technology is palpable. Yes we have pupils demonstrating increasing skills with Scratch; yes we have pupils showing increased responsibility for their learning; and yes there is a clear impact on all curriculum areas. Our investment in Google, both in time and money, is clearly justified and appreciated by our governors, parents and the wider school community.

Staff are embracing the new programming elements of the Computing curriculum and many pupils are beginning to challenge staff knowledge – a clear indicator that we are on the right track. In addition, the digital literacy elements are now of a significantly higher standard.

On reflection, we can appreciate how restrictive our prior limited resources were. Pupils from Year 4 are using Google classroom, which enables: whole-class collaboration; direct and personalised feedback; removal of many access barriers to learning; significantly-raised engagement of the pupils; greater awareness of e-safety for both pupils and adults.

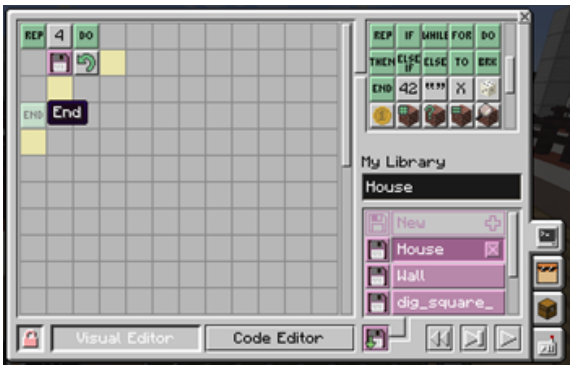
As digital natives, our pupils eagerly participate in a vast range of online activities both at school and home. Like most schools, we regularly address e-safety concerns with pupils and invite parents to take part in workshops but, equally, we are often preaching to the converted or take-up is lower than hoped for. With our increased technology in school, we have been able to improve appreciation by the pupils of online dangers. They are more aware that they leave a browsing history that can be - and is - reviewed. Pupils are starting to realise that sharing thoughtless comments online has repercussions. There is a growing awareness of their digital footprint through real experience within a protected environment not just through e-safety activities and videos. Our hope is that this better prepares them for a life lived increasingly online.

PRIMARY MINECRAFT CLUB ENCOURAGES CODING

At Gilmorton Chandler CofE Primary School in Leicestershire, Andrew Shields gave pupils a chance to extend their enthusiasm for Minecraft into different areas.

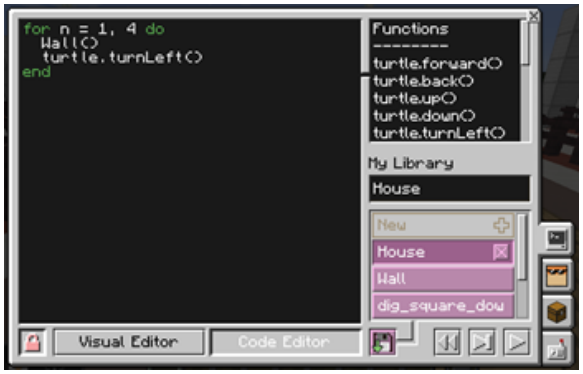
Having started a Minecraft Club this term and finding that the children I had chosen were more than competent in finding their way around the game, I wanted to give them something that would extend what they knew into different areas in order to make the club a worthwhile experience rather than just a place for them to play. What I eventually chose to do this was Computer Craft Edu (computercraftedu.com/). Computer Craft Edu is a mod that has been produced by the same company that make Minecraft Edu (a version of Minecraft developed for a school environment) and can be accessed through Minecraft Edu, although you don't need to be using Minecraft Edu to access the mod as there is a similar mod for regular Minecraft.

What Computer Craft Edu does is add programmable turtles to your Minecraft World. These turtles can initially be run using direct commands; if you press the forward arrow, it will move forward one space, etc. They can dig and they can build. In fact they can pretty much do all that a player could do but with the advantage that it can be done remotely.



Once children have mastered these simple skills (if they have used Bee Bots this is quite similar), then they can move onto a visual block based programming environment where the children can explore repetition and loops; if… else, else if… and more. With these simple tools children can quickly dig, build, create and explore a Minecraft World.

When the concepts of programming are in place to a level where you feel you can move children on, there is a text based code editor. By flipping between the two programming environments children can see similarities between the first and second formats. You can also call a program from within another program (introducing procedures). I'm excited by the possibilities and so are the children.



APPS TO HELP DEVELOP CODING IN THE PRIMARY SCHOOL

Michael O'Kane, the EU Codeweek Ambassador for Northern Ireland works for iTeach. He shares some suggestions for applications that can make an impact in the primary classroom.

In my experience, giving children in the primary classroom the opportunity to program has two major benefits:

- increased knowledge of code which will be of paramount importance in this ever increasing digital world.
- developing thinking skills, personal capabilities, problem solving, logical and computational thinking skills - key attributes which employers are on the look out for as they seek to engage creative, innovative young people.

Here are four apps I've found useful for introducing computational thinking to our young people.

Bugs and Buttons (ages 4-6) is a fantastic for early years children. Very pupil friendly, it introduces the concepts of sequencing, basic algorithms and writing instructions that underpin programming (bit.ly/1TWREGz). Children also love the superb graphics. Teachers have been impressed with how well it can be integrated into the early years' curriculum. Children get the opportunity to look at patterns, sorting and counting as well as developing problem solving skills at an early age. The teachers also enjoyed solving some of the tasks!

Scratch Jr. (scratchjr.org) allows children (6-9 years) to move away from being consumers of digital content to being *creators*. A block programming language allows pupils to create their own interactive games and stories. I have loved using this in the classroom as children express what they have learnt about their locality, looked at rainforests or space travel. Children have driven the learning in the classroom, discussing the content, characters and backgrounds and working together in groups to complete tasks.

Hopscotch (ages 8-11) is my favourite coding app! Algorithms and Debugging are terms that all budding coders will know. They underpin a lot of what we teach through the curriculum. Drawing up instructions and solving potential bugs provide tremendous learning opportunities. Alongside creating games I have used this app to look at properties of shapes and tessellations. Children program characters to move along the outline of a shape, taking into account angles, perpendicular and parallel lines - all learning outcomes covered in KS 2 classrooms (gethopscotch.com).

If you pardon the pun, the use of drones is taking off all over the country. Architects use them to obtain aerial views of sites, sports teams to profile players and give feedback on performances. I have used **Tickle** (ages 8 -11) and Parrot Drone extensively. The level of engagement of pupils is excellent. They can use the simple interface to drag blocks to fly the drone, turning it through 360 degrees, again taking account of the properties of shapes. See tickleapp.com.

PREPARING CHILDREN FOR JOBS THAT DON'T YET EXIST

When you were seven what did you want to do when you grew up? A footballer? An actress? An astronaut? If your name is Tim Peake then well done, you've made it! The chances are, though that you haven't made it to the hallowed turf in Old Trafford or on to the movie screens, but you're probably in a career that didn't exist when you were seven.

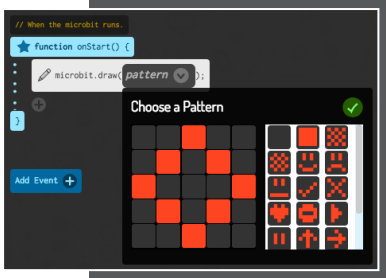
Teachers are charged with developing not only communication skills, 'writing, reading and arithmetic' but also skills to prepare young people for a future workplace where the jobs and careers they may take up in 10-15 years time are not a reality now.

One thing is certain, these future careers will involve the use of new and emergent technologies. The knowledge of how to create with these technologies will be of huge importance to those working in the future. Richard Branson recently stated, whether we are fighting climate change or travelling into Space, everything is moved forward by computers, and we simply don't have enough people who appreciate how that happens.

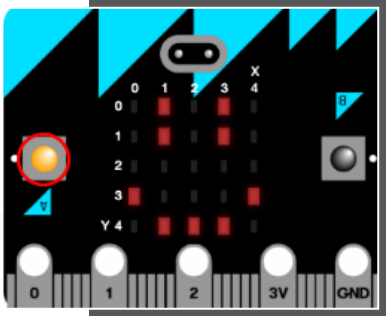
Taking part in STEM challenges prepares children for situations that may occur in many future careers in science, technology and engineering.

NEW MICRO:BIT PROJECTS FROM CODE CLUB

Code Club have produced a series of fun activities for the BBC micro:bit, which are now available to access online. Each project is intended to take about an hour to complete, and is suitable for children from KS2 onwards. The project pages link to an emulator, which means that even if you don't have a micro:bit, you can still have a go at trying out the projects.

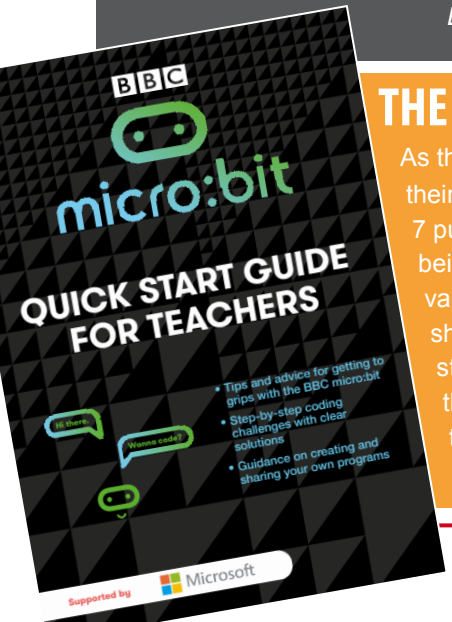


There are ideas and instructions to make an interactive badge, a reaction game and a compatibility tester to "rate your mate".



The step by step guides are easy for children to follow, and introduce the basics of physical computing using the micro:bit in an engaging way. For teachers / educators who currently run a Code Club, there will be an opportunity to apply for micro:bits via the Code Club website later in the year. In the coming weeks, the organisation will be posting news about how volunteers can apply for micro:bits. So stay tuned! For further details about Code Club's micro:bit projects, visit jumpto.cc/microbit.

Emma Norton



HOW CLASS BLOGGING HAS ENGAGED CHILDREN

Pete Jeffreys, a CAS Master Teacher at Manor Primary School in Reading reports on the interim findings of school research into the positive impact made by blogging.

We are one year into a two-year project to establish class blogging throughout our school. While there has been variation in the confidence and uptake of staff to the project, we are beginning to see how it can have an affect on a range of children across the school. Blogging has made a significant change to how some teaching and learning has taken place over the past year. The impact has varied depending on the age of children, teacher confidence, and individual children's approaches. A recent report at bit.ly/mpsblog summarises our findings.


Magpie class busy blogging

The changes noted in the report cannot be attributed solely to the blogging project, but we recognise it could have played a significant role. 20% fewer children felt that English/ literacy was their least favourite subject in 2015 compared to 2014. We hope that an enthusiasm and passion for reading and writing through new media has helped to break down preconceived ideas among pupils (and potentially also among parents). In 2014 79% of children felt that school was important to them. In 2015 this has increased to 92%. Reasons expressed varied from valuing learning, to gaining skills for future jobs and careers, and for developing friendships. Children have also shifted their emphasis about what makes a good lesson from being dominated by "fun" (38% in 2014) to an emphasis on "learning", "concentration", "listening" and "working hard" (40% in 2015).

We are proud that the efforts made by our staff and pupils are recognised, both in our partnerships schools within the project, and more widely. The school looks forward to building on these strong foundations through year 2 of the project, as we begin to look at an ongoing role for blogging as part of our learning community.

THE BBC MICRO:BIT NOW ARRIVING IN SECONDARY SCHOOLS

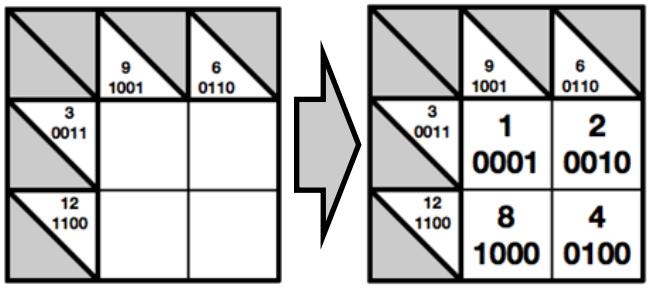
As this issue of **SWITCHEDON** was being finalised the first secondary schools were receiving their allocation of micro:bits. Each registered school will receive enough devices for each Year 7 pupil with a few extra to allow for broken on arrival and for teachers. Around 1 million are being manufactured and the roll-out is prioritising delivery to schools so they can take advantage of them during the summer term. With four different coding interfaces available, there should be plenty of opportunity for teachers to introduce projects appropriate to their circumstances. The micro:bit quick start guide is a good starting point (resources/3871) introducing the micro:bit via the blocks based IDE before developing projects in TouchDevelop, but there are now a host of tutorials on the micro:bit to help you get started. Check out 'getting started' on the website: www.microbit.co.uk.

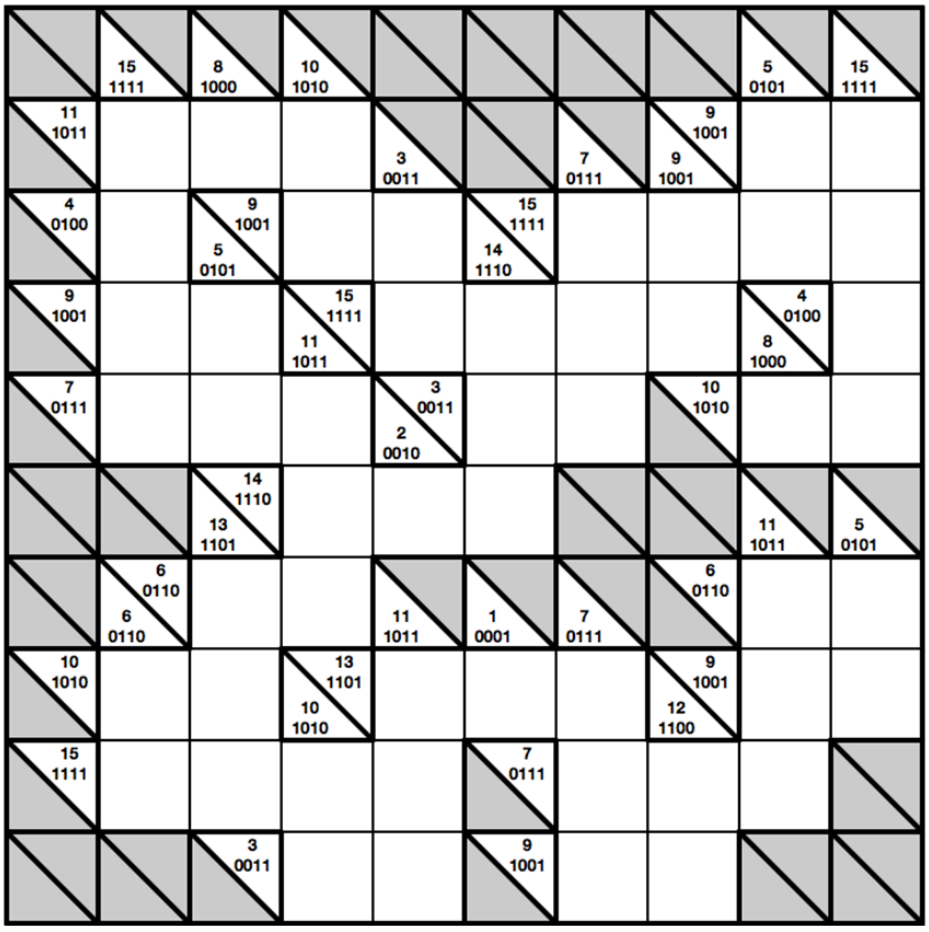
LEARNING COMPUTING CONCEPTS BY SOLVING PUZZLES

Puzzles are a fun way to introduce computing concepts and develop computational thinking skills. Paul Curzon, from cs4fn invites you to try a binary Bakuro puzzle.

Bakuro puzzles give a way to practice not only logical thinking but binary, helping to give a deeper understanding of what binary numbers are all about. The empty cells of the Bakuro grid must be filled with the numbers 1, 2, 4 and 8 (i.e. only powers of 2) together with their binary equivalent. As with the popular Kakuro puzzles the numbers in each block in a column or row must add up to the number given in the clue above or to the left, respectively. No number can be used twice within any sum. The clues are given in both binary and decimal. A simple example is shown right with the solution with a larger one to try below.

To solve the puzzle let's first focus on the top left cell. The top row adds up to 3. The only way this can be done with the numbers 1,2,4 and 8 is with 1 + 2. Now the leftmost column must add up to 9, so it must be 1 + 8. The top left cell must hold a number from both those sums, which means it must be 1. Convert the 1 to binary (0001) and we have filled in that cell. You should be able to fill in the rest in a similar way.





PUZZLE BOOK OUT NOW!

With support from Google's CS4HS programme, the cs4fn and Teaching London Computing Team, have developed a resource area on puzzles for teachers (bit.ly/1RkOuva). It includes our first computational thinking puzzle book, and downloadable version of puzzle sheets, including this one, with solutions.

Bakuro puzzles gives practice converting numbers to binary, but the link is deeper than that.

Binary numbers are just a way of making up numbers by adding powers of 2 (1, 2, 4, 8, ...), just like decimal numbers involve adding powers of 10 (1, 10, 100, 1000, ...). There is a pattern in the binary numbers. The answers that go in the grid only have a single 1 in their binary representation.

The positions of the 1s in the binary of the clue actually tell you which numbers you are looking for. 12 is 1100 in binary which says that 12 is made up of one 8 (1000) and one 4 (0100) with no 2s and no 1s, added together. The row must therefore hold an 8 and a 4. Similarly, 9 is 1001 in binary: one 8 (1000), no 4, no 2 and one 1 (0001) added together, so its column must hold an 8 and a 1. Likewise, 7 is made up of three numbers, (a 1, 2 and 4) and 15 from all 4 numbers. The puzzle illustrates how binary numbers are built from powers of 2. Sign up for more cs4fn resources at bit.ly/1UqzavU.

Teachers have access to the most effective resources in their classrooms already, whether they realise it or not! The previous experiences of pupils are useful in developing a deeper understanding of Computing topics across a variety of contexts. Constructivist learning theory considers that pupils learn in an interactive manner, building upon previous experiences. Attributed to Jean Piaget, the theory outlines that new knowledge is constructed based on what the learner already knows. When we see something new, we try to make sense of it with what we have known to be true in the past. Computing teachers sometimes neglect the links between prior knowledge and new understanding, which can have the effect of making learning more difficult than it should be.

LANGUAGES FOR LEARNING?

What programming languages are you teaching your students? We're preparing them for the next ten years - so it's Hack all the way, right? Hack is Facebook's language, now open-sourced. I confess I've never even seen it! Or we're removing barriers to access - so is it Scratch for all? Opinions abound, but what's everyone else using - and why? Have some teachers found reasons for avoiding the favourite languages that we would all benefit from knowing? Are there benefits to languages we've rejected that we didn't know about, but which other teachers can share?

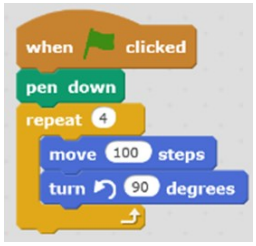
I've created an open survey to find this out: bit.ly/code-langs. Early results are already very interesting. The languages known by more than half the teachers are: Python, Scratch, Basic, JavaScript, Java, Logo, and Pascal (in that order). But which languages do they recommend for teaching? Only two of those are similarly recommended - Python and Scratch - and support for both is much lower than the number of people who know them. Is this perhaps a suggestion that we're still looking for the right modern language to teach? Or that the wealth of resources for self-teaching and adults has a long way to go to be ready for the classroom? Fill out the survey, add your comments at the end, and I'll publish everything - stats, analysis, recommendations - for everyone to read and pick over. *Adam Martin*

RECOGNISING THAT WE MUST PLAY THE LONG GAME

Pupils may know more than you think. Stuart Davison of Birmingham City University draws out the implications for teaching. It may save you planning time as well!

There are multiple threads throughout the Computing National Curriculum that appear across all of the four Key Stages, each time building upon the previous experiences of learners. For example, at Key Stage 1 pupils should be taught to understand what algorithms are, how they are implemented as programs on digital devices, and that programs execute by following precise and unambiguous instructions. Bee-Bots seem to be the de facto resource to create and debug simple programs, and for good reason. Pupils in Years 1 and 2 can pace out shapes they are familiar with for example. They can relate their movement to that of a floor robot. Programming the Bee-Bot doesn't seem quite so daunting having acted out the problem first.

So how can this be extended to KS2 where pupils should be taught to design, write and debug programs that accomplish specific goals and where there are quite specific requirements to become familiar with programming constructs and approaches. Building upon the learning experiences of pupils provides an opportunity for them to learn new knowledge based around a familiar context. If Bee-Bot is synonymous with KS1 teaching, perhaps Scratch should be considered the same at KS2. Block based environments give learners a sound basis for programming, dragging and dropping shapes whilst not having to worry about syntax. The progression from buttons on Bee-Bot to ordering instructions on a screen is aided by providing a familiar, previous context. Constructivists appreciate the importance of drawing a square again. Yet this is just the start. Familiar methods applied to new contexts help build confidence in learners. In turn this allows them to develop new understanding and extend what they can do. Having used a linear sequence of instructions, you could progress to using loops, repeating blocks of instructions; then variables to store the number of sides for a shape and so on.



Secondary teachers can also look to build on these experiences as a starting point for later development. At KS3 pupils should be taught to use two or more programming languages, at least one of which is textual. In Python pupils can import a turtle module that replicates the functionality of both Bee-Bot and Scratch. Why not start pupils with a program that draws a (yes, you guessed it) square? Again, previous experiences are used to assimilate new information. Explicit links between blocks in Scratch and the equivalent Python instructions can be made. The learning moves on without having to explain afresh a new context. It is not expected that Computing should become a shape drawing class, but basing current teaching on prior experiences makes the assimilation of new knowledge easier and aids progression of learners.

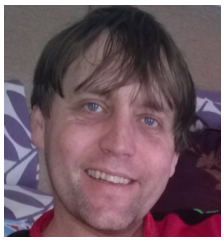
```
from turtle import *

floor = Screen()
floor.setup(500,500)
piaget = Turtle()

for i in range(4):
    piaget.forward(50)
    piaget.left(90)

floor.exitonclick()
```

WRITING PROGRAMS TO HELP IMPROVE MATHS

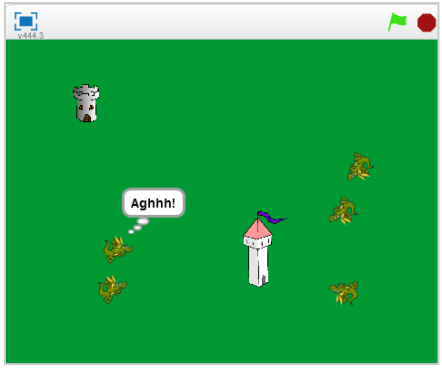


David Hill and Simon Marsden from the University of Portsmouth share ideas for coding to improve mathematical understanding.

Why not use programming to help children improve their understanding of maths (and by association science)? We get them to calculate things by hand because we want them to understand the processes, but real understanding comes when the children can see the maths in action. Games rely heavily on mathematical procedures to make things behave in a specific way on screen, so why not use games to add real context to maths?

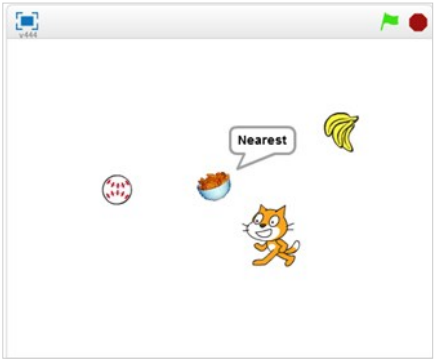
If the focus is on applying the maths, then we don't want to obscure this by having to understand new programming elements. One approach is to use learning frames in the way one might in English lessons. The main part of the program is provided. What stops it working as intended is some element of maths.

How might this work in practise? Let's take an example of a tower defence game in Scratch (see bit.ly/1RvbNhT).



The goal is to stop waves of enemies reaching your castle, by placing gun towers on the map, which shoot the enemies as they pass. With so many enemies, how does the tower know who to shoot at first? We could say shoot at the nearest one; but how do we calculate which is the nearest? The challenge is to program Scratch blocks that can use the information they have in the code (the coordinates of the sprites) to calculate the shortest distance. Drawing out a simple scenario, with one enemy at, for example, (10,10) and the castle at (2,4), allows pupils to see the change in the X and Y axis between the two points. Drawing the resultant distance creates a right-angled triangle. Children can then explore how you work out that distance using Pythagoras, with the formula being built up in stages using blocks i.e. it is decomposed for them.

Testing code in a complicated game might cause confusion, so first test by using a simple example in Scratch, where pupils can see which is the nearest object (see bit.ly/1MLvsIt).

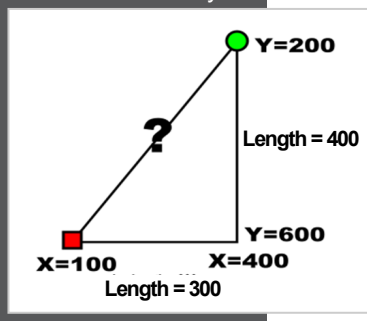


Once the code works, the children can copy it to the actual game using the Scratch backpack feature, generalising the problem in the process.

This is a simple example but consider the possibilities of such an approach, from Key Stage 2 maths to more complex areas. such as planetary motion (sin and cos at KS3/4), or the ballistics that Adrian Oldknow shared in the last issue of SWITCHEDON.

THE POWER OF FUNCTIONS

All undergraduates at Portsmouth learn how to use Pythagoras to measure the distance between objects on a screen, as David has outlined in the article (left). The only difference is that we use Python and / or Java to do this. Once they understand hopefully familiar maths in this new setting, we obviously want them to reuse the code. It won't surprise anyone familiar with Python that we use functions (or 'methods' as they are known in Java) to do this.



This is where user-defined blocks come in. They are a fantastic addition to Scratch 2 and work in both the off- and online versions. A block can be directly equated to a Python function. The block holds the code that you want to repeat. In Python a Pythagoras function might look something like the code at the bottom (it can be improved of course, but the structure makes the process clear). The function would be called using `pythagoras (100, 400, 200, 600)`

Having the bulk of the game provided, so that pupils can concentrate on the specific programming problem, is a practical example of both abstraction and decomposition. Of course, if the problem is solved in the simple example, they can then use the same code in the more complicated game by recognising the pattern and generalising.

```
import math

def pythagoras (aXpos, bXpos, aYpos, cYpos):
        hypoteneuseSquared = (aXpos – bXpos)*(aXpos - bXpos)+(aYpos - cYpos)*(aYpos - cYPos)
        hypoteneuse = math.sqrt(hypoteneuseSquared)
        return hypoteneuse
```

COMPUTER SCIENCE FOR ALL?

Following the introduction of the new Computing curriculum I have often been asked if it is appropriate to teach programming and computational thinking to pupils with special educational needs, in particular those working below National Curriculum level. It is clear that for those young people with profound and multiple learning difficulties it would not be appropriate or meaningful, but there are benefits for pupils working around P5 and above.

Computational thinking helps pupils to learn essential problem-solving skills that can be applied across the curriculum: sequencing events, breaking down problems into smaller parts and identifying patterns. Pupils working below National Curriculum levels can be taught these concepts through a range of activities that also meet individual priorities, for example improving communication, social skills and life skills.

Computational thinking skills are equally valuable for pupils with SEND in mainstream schools: for example decomposing complex problems to make them more manageable; detecting and correcting errors. Text-based programming languages can present barriers to those with poor literacy, and some pupils may require model code to adapt and build upon. They might also benefit from the immediate feedback of physical computing devices.

There are a number of places where teachers can find guidance, ideas and resources for teaching Computer Science to pupils with special educational needs. CAS #include, a Computing At School working party to help all students access Computer Science, has links to resources and upcoming CPD events for teachers on its website: casinclude.org.uk. It is also the home of the Revised P Scales for Computing, a piece of work to update the P Scales statements to better reflect the content of the new Programs of Study. Most teachers will be aware of the excellent collection of lesson ideas and teach-yourself resources on the Barefoot website. What you may not know is that there is a growing number of resources created specifically for teachers of pupils with special educational needs and disabilities under the *Teach SEN* tab. *Catherine Elliott*

MATHEMATICAL MUSINGS AND EXPLORATORY CODING

Mark Thornber, from Durham Johnston Comprehensive School, kicks off a regular series exploring ideas from mathematics that lead to interesting coding tasks.

$$8\overline{)5{\cdot}5^{5}0^{2}0^{4}0^{0}} \quad 0{\cdot}625$$

In this first column we will look at a mathematical idea that every pupil should have met in Primary school – converting fractions to decimals. This simple idea can lead to a coding challenge in Python (or your own favourite language). But first the maths. To convert a fraction to a decimal, you just divide the numerator by the denominator using short division. For example, consider $^5/_8$. This gives a terminating decimal 0.625, because the final remainder is 0, but change to $^5/_7$ and we get a recurring decimal, 0.714285714285... with a repeat of length 6, because the remainders start to repeat. We can get more complicated behaviour where the repeat does not start immediately after the decimal part, for example:

$$\frac{47928}{725} = 66.10\dot{7}586206896551724137931034482\dot{8}$$

The dots above the 7 and 2 indicate that only the last 28 digits repeat. How did I get that last monster? I wrote a program in Python to mimic division of course! The maths is Primary school, but we do need a range of programming ideas to construct a solution. We can build up the answer as a string by successively calculating each digit and concatenating. First get the whole number part and the remainder:

```
whole = numerator // denominator
remainder = numerator % denominator
```

We now get each new digit by repeatedly multiplying the remainder by 10 (the equivalent of writing it next to a zero in the handwritten version) and then dividing by the denominator again. There are two ways to end: either we get a remainder of zero, as with $^5/_8$, and the decimal terminates, or we get a remainder that we have seen before and then the decimal repeats. To check for this we need to keep track of remainders and stop dividing when one of these possibilities occurs. My solution:

```
remainders = [remainder]
repeating = False
while remainder > 0:
    digit = (10*remainder) // denominator
    remainder = (10*remainder) % denominator
    answer = answer + str(digit)
    if remainder in remainders:
        repeating = True
        break
    else:
        remainders.append(remainder)
```

To finish, we must solve the problem of printing out the answer. It's not easy to put a dot above a digit in IDLE so I use string operations to surround the repeating part in brackets, e.g. `0.(714285).` Perhaps you have a better idea?

Mark began his career as an academic pure mathematician but quickly decided that was not for him. He has been a maths teacher at Durham Johnston Comprehensive for over 20 years. The rebirth of Computer Science allows him to indulge a passion for links between maths and computers, teaching Computing and contributing to CAS forums on a regular basis.

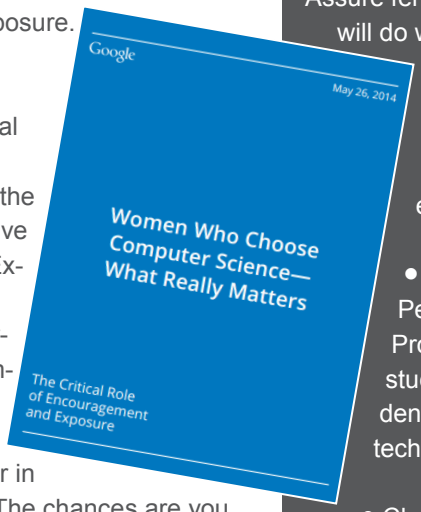
ADDRESSING THE GENDER DIVIDE STARTS EARLY IN SCHOOL

Major companies recently released breakdowns revealing low percentages of women in technical jobs. Emma-Ashley Liles, Industry Liaison for CAS #include suggests ways to address the problem.

The tech industry is slowly admitting it has a diversity problem. The wider implications of figures like 17% female engineers at some of the world's largest and influential companies is not just a problem for women. People with tech skills are quite rare. I know this, because I am a software engineer at a music streaming company. Involved in the hiring process, I know the struggle of finding good staff - a struggle that is only getting worse. In the next decade the UK will see a skills shortage of 1m tech workers. That gap could be filled by women and companies would be better for the different views and experiences they would bring. If you don't believe me, ask yourself why the Apple health app doesn't track periods.

Figures in the UK for young women gaining Computing qualifications show the problem starts early. 23,590 female compared to 29,607 male took ICT GCSE in 2012. 44% women - not bad but not Computing. Girls did better than boys with 79% A-C pass for women and 71% A-C pass for men. When it comes to A levels, 4,284 female compared to 6,804 took ICT in 2012. But 3,513 male compared to 297 female entrants for Computing! 297 is probably fewer friends than you have on Facebook. I give up the majority of my vacation days and countless weekends to mentor young women interested in computing and I have probably personally tutored at least 10% of these women. Further on at university the figures are a little murkier and difficult to decipher, but somewhere between 12% - 18% of all CS degrees are awarded to women.

What can we do about this? It just so happens that Google have already set their mind to this. #include were lucky enough to be in the audience when Google revealed the findings of their in depth study across the whole of America into the reasons behind these figures. The study is called "Women who choose computer science ". (bit.ly/1Gzew4h). One significant point was that factors that influence girls are already present before they choose GCSEs. They suggest we can make a difference by looking at the two simple things that matter the most - Encouragement and Exposure.



Encouragement means non familial encouragement which also helps the girls build a positive self perception. Exposure comes in the form of opportunities to try computing for themselves. Picture a software engineer in your minds eye. The chances are you saw a sterotypical antisocial nerd. Girls can see that too, and it needs to change for this problem to be solved. In the words of Geena Davis, arguing for positive role models, "If she can see it she can be it".

This is all well and good but we also need some practical advice. Luckily, the wonderful team at the National Centre for Women in Technology (NCWIT) have a wealth of resources available online.

national center for women & INFORMATION TECHNOLOGY

Here are my top 5 tips from the NCWIT resources section:

- Know your audience

Be aware of the toxicity of the nerd image. Girls don't even have to be studying CS to perceive that negative image! Be inclusive with the language and images you use.

- Confidence can impact ability

Assure female students that they will do well. Be aware how skewed your female students self perception of their abilities truly is. Maths is not an essential!

- Encourage collaboration

Peer Learning is like Pair Programming. Female students feel more confident and it is a recognised technique in industry.

- Challenge isolation

Applies to any underrepresented students. There are many initiatives to connect students to encourage and support each other

- Know your role models

Cited as one of the major barriers. Look for people they can look up to right now. My personal heros are Anne-Marie Imafidon, Carrie-Anne Philbin, Cate Huston and our newest addition at work, Janey. She marks the point at which our tech team Ihas reached 25% women. Let's get that up to 50% so we can see better health apps and hopefully, a better world.

#include computer science for all

CAS #include are passionate about giving all students the opportunity to study Computing regardless of gender, race, socio-economic status, SEN or disabilities. Want to help? casinclude.org.uk.

COMPUTER SCIENCE IN SCHOOLS: A MOVEMENT GROWS ACROSS THE GLOBE

Computing at school is a hot topic in dozens of countries around the world. All the core issues are the same: the idea of computer science as a foundational discipline, computational thinking, the role of programming, the digital skills gap, and so on. CAS Chair, Simon Peyton-Jones surveys the landscape.

INTERNATIONAL OUTLOOK FROM THE VERY START

From its very outset CAS has benefited from the vision and leadership of others across the globe. I remember three in particular from the early days.

First the Computer Science Teachers Association (CSTA) in the USA has been established far longer than CAS, and has produced a series of articulate, well-evidenced policy papers making the case for computer science as a school subject, and an outline curriculum suggesting what it might look like. Their 2010 report, 'Running on empty: the failure to teach K-12 computer science in the digital age' is a good example. Also from the USA, Mark Guzdial's blog posts are an ongoing source of insight for me.

Second, Tim Bell and Mike Fellows's inspirational work on Computer Science Unplugged brilliantly articulated the idea that computer science is not only (or even primarily) about technology, by the simple device of not *having* any technology. They also opened our eyes to the idea of computer science as a *primary* school subject. And they are still at it! If you have not drawn on the material in the Computer Science Field Guide that Tim and his colleagues have been working on over the last few years, you should drop everything and take a look.

Finally, led by Judith Gal-Ezer and others, Israel was way ahead of the game: they undertook a major review of computing at school in the 1990s, and since then computer science has been well established in upper secondary schools. There is much to learn from their experience; for example, Computer Science Concepts in Scratch (Armoni and Ben-Ari, 2013) is an excellent free online book (bit.ly/1ULtj5Z).

As the sidebar shows, there is much we can learn from each other and, at a practical level, there is a rich cornucopia of resources available from around the world that we can use in the UK, and vice versa. The UK has moved particularly quickly, but there are now signs of movement everywhere. Here are some particularly prominent examples.

In January 2016 President Obama announced the "Computer Science for All" initiative, backed by $4 billion. (Yes, that's four *billion* dollars.) In his 2016 State of the Union address, Obama said "*In the coming years, we should … offer every student the hands-on computer science and math classes that make them job-ready on day one.*" The USA is particularly complicated because education is mostly in the hands of 50 states and thousands of school districts, rather than the federal government, so national initiatives have many moving parts.

Meanwhile the CSTA and code.org are coordinating a major collaborative endeavour to develop a K-12 computer science curriculum. They are explicitly inviting contributions from abroad to help articulate exactly what it is that we want our children to learn. I expect the outcome of this process to be very useful for us too.

In December 2015 the Australian government announced A$51 million to "*help Australian students to embrace the digital age and better prepare for the jobs of the future*". This builds on Australia's brand-new national curriculum for digital technologies, which "*focuses on developing foundational skills in computational thinking and an awareness of personal experiences using digital systems.*"

Across Europe there have been a series of recent high-profile reports with action-oriented titles such as "*Informatics education: Europe cannot afford to miss the boat*" (2013), "*Teaching computer science in France: tomorrow can't wait*" (2013) and "*Computing in schools: a call for action from Informatics Societies*" (2014).

ECONOMIC IMPERATIVES

As technology becomes ever more pervasive, traditional trades are disappearing. The world of work has become more globalised, more collaborative and thus the skills demanded by employers are shifting. So how can education best prepare young people to navigate their way through an increasingly interconnected and complex world? A recent report from The Economist Intelligence Unit '*Driving the Skills Agenda: Preparing students for the skills of the future'* makes interesting reading. The top level finding that "50% of teachers, students and executives cite problem solving as the most important skill for potential employers, with 70% expecting its importance to increase over the next 3 years" provides the compelling backstory to the need for curriculum change. Full report: bit.ly/1FXrtbp.

On a smaller scale, we've been in discussion with enthusiasts and policymakers in dozens of countries: Argentina, Australia, Bahrain, Belgium, Brazil, Canada, Denmark, Estonia, Finland, Germany, Greece, Hong Kong, Hungary, India, Italy, Kazakhstan, Japan, Latvia, Malaysia… and that's just the first half of the alphabet. Moreover, although CAS is UK-centric, nearly a thousand of our 21,000 members hail from outside the UK – and there are even a few CAS hubs abroad. Read more about our hub activities in Asia for example on p16.

Although much is going on elsewhere, the UK is a particular crucible of change. England is so far the only country in the world that has established computer science as a foundational subject that *all* children learn (as they do natural science and mathematics), from primary school onwards. I am hugely proud that CAS won the 2014 Informatics Europe Best Practices in Education award, *for "its outstanding achievements which resulted in computing being officially established as a mandatory subject for primary and high schools in the national curriculum of England"*. This is a marvellous measure of our collective achievements. The Raspberry Pi is already an international phenomenon, and I'm pretty sure that the BBC micro:bit will be too.

While our international colleagues are interested in our curriculum, I think that battle is all but won. Instead, discussions often focus more on the CAS community itself (how did we build such an effective grassroots community, that alongside teachers includes software professionals and university academics?), and the question of training and implementation (online resources, the NoE, hubs and so on).

That doesn't mean we know all the answers! Far from it; we are in the midst of an exciting but turbulent implementation that will last a decade. And we still have so much to learn from our colleagues overseas. But it does mean that what happens in this country matters well beyond our borders. The world is watching us with intense interest. We have so much in common: let's look outward as well as inward, encourage, share, and learn.

SUPPORTING AND NURTURING COMMUNITIES OF PRACTICE

Computing At School started out in 2008/9 as a small 'guerrilla group' arguing the case for curriculum change. Over the next few years we found we were tapping into a more widespread mood. The desire for change was apparent amongst, not just teachers, but employers and academia too. The world was changing, at a faster pace, and many agencies found they were grappling with the same conundrum; how to prepare children to solve problems that haven't yet even registered as problems.

From such high level aspirations come practical considerations. Educational change comes slowly, as any ICT teacher, trying to embed the use of new technology in their school will testify. Many of these ICT teachers came to the subject via circuitous routes. Most have no background in Computer Science, but it is doubtful if any other subject can tap such innate enthusiasm.

What makes our curriculum journey so interesting to those outside the UK is the success of a twin track approach; lobbying for change at policy level driven by a truly grass roots movement. Sometimes messy, at times chaotic, CAS recognises the need for change, but also has an ethos that determines to do something about it. Through local CAS Hubs and the Network of Excellence, Computing teachers have become a beacon of excellence that the world increasingly looks toward for inspiration and answers. Google, in recognition of our success recently provided seed funding to initiate similar groups in France, Spain, South Africa and India. We look forward to working with them.

The inclusivity, sense of common purpose and willingness to 'walk the walk', rather than 'talk the talk' is a model to which others aspire. You don't need to be an expert to drive educational change, but you do need a desire to learn and willingness to share. Those who have contributed to the CAS mission can reflect with real pride on how their efforts are truly changing the world.

Simon Humphreys

GLOBAL COMPUTING: LEARNING THE LESSONS FROM INITIATIVES ABROAD

There's a global movement to teach children to code, and through this to give them tools to solve problems and understand systems. Much is happening that we can learn from. Countries are tackling the challenge in many different ways. Miles Berry explore the ways they are responding to the opportunities.

OPPORTUNITIES OF EXTRA CURRICULAR APPROACHES

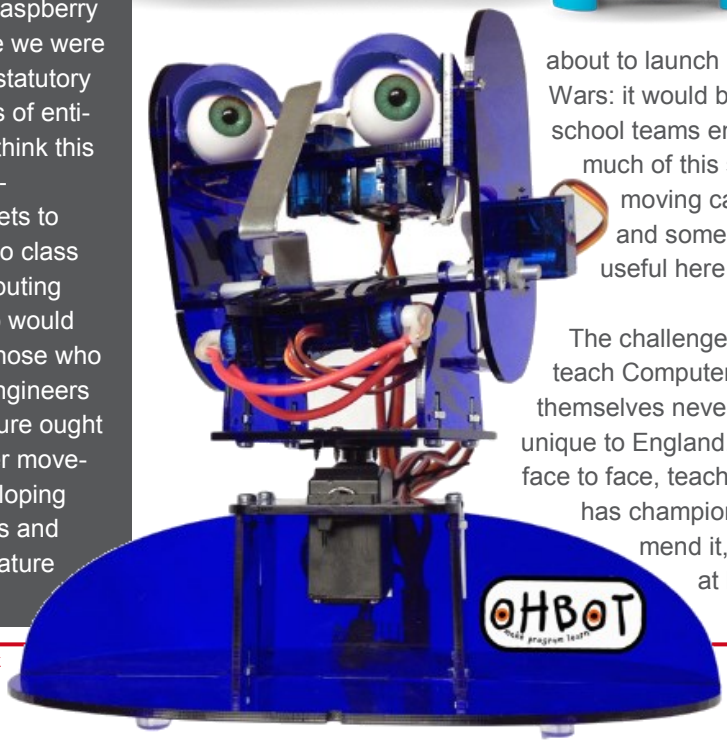
Even where computing hasn't made it onto national curricula, there's much interest in supporting provision beyond the school curriculum. It's easy to forget that Scratch itself started in the context of provision outside school, with Resnick's Computer Clubhouse work, which always had a focus on otherwise underserved communities. I'd argue that most exciting and innovative Scratch projects are coming from individual Scratchers and Scratch Studios rather than school or classroom projects.

Coder Dojo, Code Club and Apps for Good operate internationally, and there's global interest in products such as the Raspberry Pi and the BBC micro:bit. I'm sure we were right to include Computing in the statutory curriculum as this enshrines ideas of entitlement and inclusion, but I don't think this should be at the expense of extra-curricular opportunities. No one gets to Carnegie Hall just through going to class recorder lessons. Whilst our computing lessons open up CS to many who would otherwise not have studied this, those who are destined to be the software engineers and computer scientists of the future ought to be getting involved in the maker movement, taking some MOOCs, developing apps or games, going to hackdays and other things that are unlikely to feature in the provision made by schools.

England is leading the way worldwide through including Computer Science as part of its national curriculum from primary onwards, and there seems ever increasing numbers of countries that are following in our footsteps, or at least taking a keen interest in what we're doing here.

Whilst there are plenty of schools in the UK that are working with robotics and other aspects of physical computing, I worry at times that we ought to have placed more emphasis on this in the curriculum. Elsewhere, there seems much more focus on this as a vital part of computing education. ProgeTiiger in Estonia fund 80% of the cost of robotics kits for elementary schools. There's similar generous funding for robotics projects in Portuguese schools, both primary and secondary and robotics plays an important part in extra-curricular computing in Singapore and South Korea - often going way beyond the sort of line-tracing / path-following projects we see at home.

Back home, there's been more interest in this of late, with the new WeDo 2.0 from Lego and OhBot's robotic face (which links nicely with Paul Curzon's famous facial expression unplugged activity). Dot and Dash captured interest at BETT, with a default programming language that owes more to finite state machines than block based coding. Kitronik have developed a BBC micro:bit powered robot kit, and the BBC are about to launch a new series of Robot Wars: it would be great to see some school teams enter this. That said, much of this seems a step up from moving cats around a screen, and some targeted CPD might be useful here.



The challenge of training teachers to teach Computer Science, when they themselves never learnt any, isn't unique to England. The locally trusted, face to face, teacher-led model that CAS has championed has much to commend it, but it's worth looking at how other countries are approaching this. I

EARLY YEARS INSPIRATION FROM FINLAND

The importance of laying the foundations of computational thinking through the Early Years Foundation Stage featured in the previous edition of SWITCHEDON, but, for me, the most exciting work in this area is Linda Liukas's inspiring Hello Ruby project in Finland.

Hello Ruby, the book, was funded via Kick Starter: it's a beautifully illustrated children's story about Ruby, who loves learning and hates giving up, and her quest to collect gems in a fairy-tale world populated by snow leopards, penguins, fire foxes, androids and Django and his pet python. Linda doesn't attempt to teach pre-schoolers to code through the story (thankfully), but she does get them thinking in an unmistakingly computational way.

There's a companion website, at www.helloruby.com, with some brilliant, play-based ideas such as a cut out and keep computer (cheaper even than a Pi Zero!), and stick on power buttons to link everyday objects (at least in a child's imagination) into the internet of things. Diane Levine is running a session at Roehampton University's Festival of Computing looking at how we might draw on these ideas at home. Linda has also been helping with the development of Finland's new computing curriculum, which comes into effect in September - check out the support materials she's produced at koodi2016.fi (using Google translate if you can't manage the original Suomi!).

don't doubt that a significant proportion of the $4 billion promised by President Obama to support CS for All in the states will be spent on training and professional development for teachers.

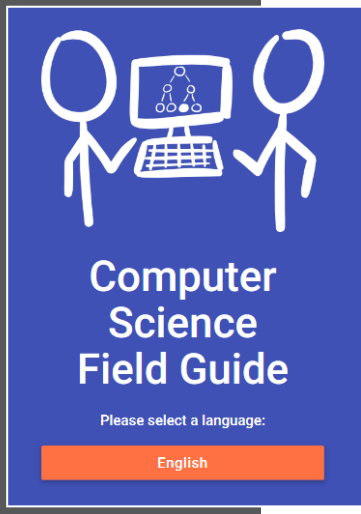
In terms of teachers' own subject knowledge, the most significant contribution is undeniably the MOOC (massive open online course) movement, with very high quality teaching materials made available for free online by prestigious universities, alongside facilitated online communities and assessment. Many MOOCs offer much for teachers wanting to learn more computer science than just that needed to teach the course - MITx 6.00.1x, introduction to computer science and programming using Python, Berkeley's Beauty and Joy of Computing and Harvard's CS50x are all quite brilliant general introductions, pitched at first year undergraduate level. The latter two now exist as 'advanced placement' courses for US high school students, and could be a viable option for a sufficiently motivated student who finds herself in a school without a CS teacher. Microsoft are bringing a teacher version of CS50 to London in May - see bit.ly/cs50ldn for further details.

None of the above should be taken as a criticism of how we're going about things back home, but more as some interesting, alternative perspectives on what is now a worldwide movement. These are such exciting times, not just for us and our pupils, but for teachers and students across the globe.

NEW ZEALAND CS FIELD GUIDE

The CS Field Guide (csfieldguide.org.nz) has been developed over the last few years to support the new senior high school CS standards in New Zealand. It is open source and available for anyone to use. Currently focussed on topics needed in NZ schools, we have plans to expand it to cover topics from other curricula.

This (New Zealand) summer we revamped the structure of the CS Field Guide so that it provides a much better platform for growth and internationalisation. We've also added a lot of new material, with more in the wings to roll out this year. To support growth, the entire project is now available on GitHub so you can grab your own copy and update it or serve it up locally if you want; in particular, it makes it easy for people to contribute improvements. To support internationalisation, all NZ-specific material has been moved to a "curriculums" section, so that your students can use the chapters without being distracted by information about NZ assessment requirements!



We are also starting a project to produce a new version of the sister site, csunplugged.org, so that it is more focussed on school lessons and computational thinking. CS Unplugged has been around for over 20 years, and was originally designed as an outreach tool, at a time when having these topics in the classroom seemed a long way off. Now that this has become a reality, we'll be collecting best practices will be revamping the material to be more suitable for the needs of primary school teachers, although it's great to see that lots of people have already been adapting it from the current form. "Unplugged" is also open source, so copying and editing it is encouraged. We also like it if you share it back with the community! *Tim Bell*

INTERNATIONAL SCHOOLS AT FOBISIA CONFERENCE

For two days in October 2015, Harrow International School Bangkok held the first ever FOBISIA academic Computing conference entitled "Developing an Effective Computing Curriculum". Over 30 primary and secondary teachers from all over Asia took part in what was to prove a very inspiring two days of workshops and discussions. The topics discussed were how to shape the new computing curriculum at KS3 (Prep to Y9), choosing the correct exam boards at IGCSE/GCSE and A Level, assessing computing in the classroom and how to encourage more girls into Computer Science.

The workshops covered Python development at senior school level, as well as Scratch for younger students. We also looked at the latest technologies available for the classroom, including using a Google Cardboard, which each participant got to take home with them. The conference finished with programming Python in Minecraft – a workshop that proved to be very popular!

In an ever changing subject with many international schools moving away from an ICT based curriculum to a more Computing based subject, this conference proved to be very useful for teachers and gave everyone a chance to see what is happening at other schools. This will hopefully become an annual event from now on.

Carl Turland

The Korean news service reports a fact finding visit of teachers to London schools and CAS in February

REACHING OUT TO TEACHERS IN SOUTH EAST ASIA

Building on the success of the links forged between CAS members in international schools, James Abela reports on initiatives to strengthen links with local communities.

In South East Asia, CAS has been successful in establishing a presence in international schools, making very good use of existing networks. In Bangkok, Simon Aves has been leading the way with the local ISTEC Bangkok group. Carl Turland has been organizing the teachers at international schools Computer Science forum through FOBISIA (see left), and we have been on social media to drive the message forward. Carl Turland and Simon Carter have been dealing with the AQA syllabus, whilst Harvey Taylor and I have focused on the CIE syllabus.

In Malaysia, we were a part of the ASEAN summit which President Obama attended in which coders from across South East Asia participated. In the competition, we used Live Code for developing apps and the Raspberry Pi for electronics. It was very pleasing to see an all-girl team and my A level students win top prizes. It was great publicity for Computing and very exciting for students to be involved in two hackathons. Unbeknown to most, electronics is Malaysia's top export; a message that needs to be reinforced if we are to encourage students to study Computing. Cambridge University has also been active in promoting Computer Science, with awards given for Computer Science AS levels for the first time. The region has also seen a growing trend of hackathons. Hackerearth kindly offered to host the region's first school-based Hackerthon with strong entries from a wide range of international schools. For Primary, Tanglin's Glenn Malcolm is hosting a Scratch-based competition, with students linked via video conference (see bit.ly/1oHjIk6).

The next challenge is to get closer to the local schools, and the first step towards this has been through the Google Educator Groups. Malaysia is leading the charge here with an intention to integrate Computing into their upper Secondary curriculum by 2017. I recently provided training to enthusiastic Malaysian teachers. There are willing professionals who are eager to help from a wide range of backgrounds, and now the challenge is to connect all the threads together! In South East Asia we are at the very beginning of our journey, but we hope to replicate the success that has been seen in the UK.

RESPONDING TO OBAMA'S CALL: THE FIRST USA PICADEMY

In the summer of 2015, President Obama made a call to create a Nation of Makers within the United States. Marc Scott outlines the response from the Raspberry Pi Foundation.

In response to President Obama's call to action we made a commitment: we would train 100 teachers in digital making with Raspberry Pi in 2016. In March, we made a giant leap towards fulfilling that commitment with our first Picademy in the United States. In fact, it was the first ever Picademy held beyond the borders of Britain.

We invited 40 educators to the Computer History Museum in Mountain View, CA to learn about the Raspberry Pi, Python, Scratch, the Sense HAT, the Camera Module, working with motors, buttons, switches, LEDs, and our philosophy around digital making for education. This cohort travelled from all over the United States to become the first set of Raspberry Pi Certified Educators to be trained on US soil. They included classroom teachers, librarians and community educators. This was the largest Picademy ever!

We could not have done it without a lot of support. All of us at the Raspberry Pi Foundation would to thank: Dexter Industries, Adafruit Industries, Sparkfun Electronics, and Low Voltage Labs for equipping the teachers with products and discounts; David Saunders, a Raspberry Pi Certified Educator who joined us to help participants and to speak about his school's "Labrary"; Sonia Uppal and Mark Pavlyukovskyy for speaking about their amazing work with Raspberry Pi.

Most of all, we'd like to send a huge 'thank you' to everyone at the Computer History Museum, especially Kate McGregor, Stephanie Corrigan, and Lauren Silver. CHM was the perfect venue for a Picademy in the Bay Area, since much of the history of Computing took place nearby. It was amazing to have such a rich historical context for their work with Raspberry Pi.

BEBRAS FOUNDER RECOGNISED IN EUROPEAN AWARDS

Nothing better epitomises the international movement to promote Computational Thinking than the annual Bebras competition. The idea of Bebras was born in Lithuania, by Prof. Valentina Dagiene from University of Vilnius. The activity of beavers was symbolic of the challenge… a persistent endeavour for perfection, beavering away to reach the target. Bebras is a Lithuanian word for 'beaver'. Since it's launch in 2004, it has grown to encompass 45 countries, including the UK, involving nearly a million students annually. In recognition, Valentina has been awarded the European Celebration of Women in Computing, Ada Lovelace Computing Excellence Award; see bit.ly/23Ff21r.

Bebras UK launched three years ago. This years' attracted 40,000 entries across all age groups. The challenge is a series of multiple choice online questions requiring no previous knowledge. It takes 45 minutes and is automatically marked. See challenge.bebras.uk to try it! Certificates of participation are issued. Now with sponsorship from Oxford University the highest performers in each age were invited to a finals day at Hertford College last January. Registration is now open for next years' competition.

NORTHERN IRELAND TEACHERS VISIT CAS

Excitement is building in Northern Ireland as the work of DigiSkills NI starts to impact on schools. The DigiSkills project emerged when Irene Bell, the Chair of CAS (NI); Sheila Fleming, Invest NI industry representative; and Meabh McCaffrey-Lau from NI Screen joined forces. The single goal was to raise the profile of Computing and digital technology at all levels within the NI curriculum, to set up channels through which teachers can receive CPD training. As part of the project (thanks to funding from the Department of Culture, Arts and Leisure), some teachers spent two days in London learning about CAS Master Teachers and visiting a Digital School House school.

Thanks must go to Simon Humphreys who has given freely of his time to support DigiSkills in moving the agenda in Northern Ireland forward. A further 24 teachers are currently undertaking a three-day CPD training course in coding at Stranmillis University College. Ann O'Neill is introducing the teachers to computational thinking and they are really enjoying 'playing' with WeDo and Dot & Dash Robots.

A report sponsored by Invest NI will be released imminently setting out the state of play of Computing in schools and making recommendations on how to support teachers in Northern Ireland. We await the next steps with anticipation and a sense that a wave of change is happening. We have fingers crossed that we will be successful in our funding bids and our work with government agencies, and we look forward to bringing the teachers of Northern Ireland more good news for the next academic year. *Irene Bell*

GPIO ZERO: PHYSICAL PYTHON ON THE PI

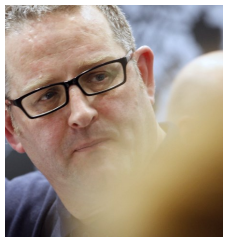
Physical computing is one of the most empowering ways of teaching computing concepts. The Raspberry Pi Foundation have created a new Python module providing an easy way to get going with everyday physical components such as LEDs, buttons, buzzers, motors and a range of sensors: it's called GPIO Zero.

GPIO Zero was designed for education, and makes physical computing projects more accessible. It provides a simple interface to components regularly used in projects. The boilerplate code needed to get going is minimal and the function names designed to be obvious. Despite the simplicity of the code required to do things with motors and lights, there are unlimited ways the standard recipes can be used, including reaction games, stop motion animation, a musical box, tweeting toys, remote controlled robots, greenhouse monitoring kits and more.

GPIO Zero can bring computing concepts alive with Python. Learn about data structures with a line of coloured lights for example, or algorithms by instructing a robot through a maze or creating a traffic lights sequence. You can investigate threading by getting LEDs to flash in the background or learn about how security light systems work with a light sensor and lights. Why not use a full colour RGB LED to learn how colours are formed and even try controlling a home-made sous vide cooker with a temperature sensor! With a little imagination the possibilities are endless.

Ben Nuttall

CONTROLLING DEVICES USING SCRATCH ON THE R-PI

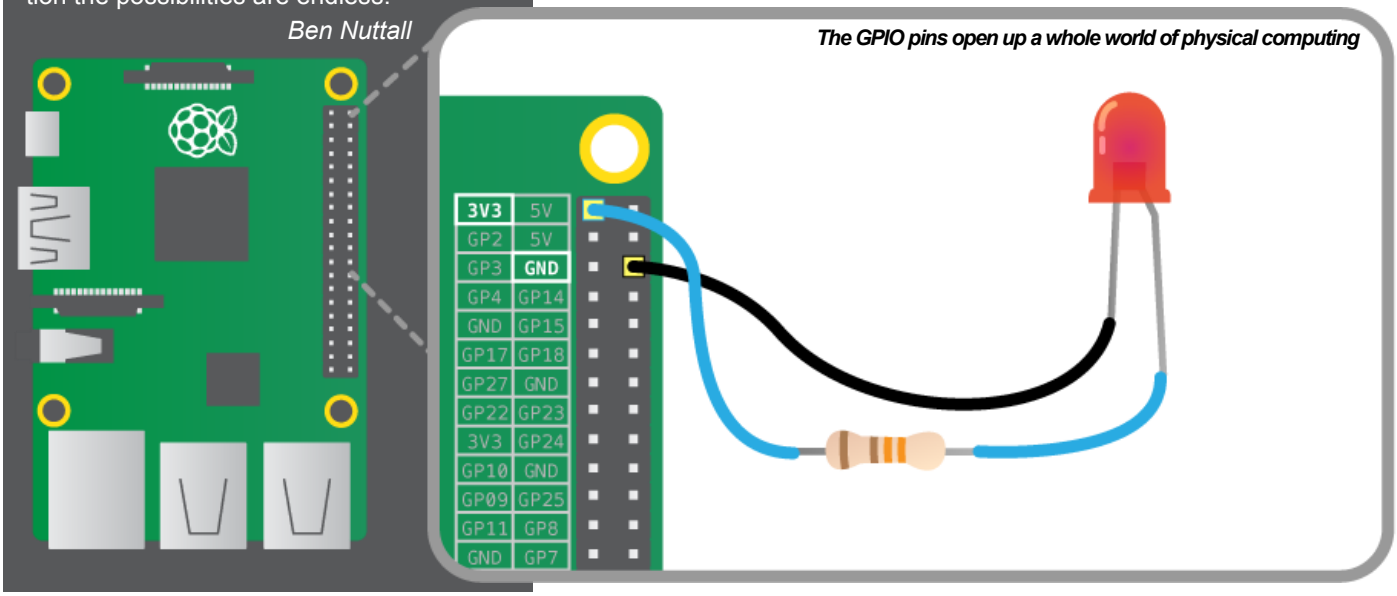


Clive Beale, of the Raspberry Pi Foundation, outlines an exciting development that allows Scratch programmers to control external devices via the Raspberry Pi.

Scratch needs little introduction around these parts—it's the *de facto* tool in computing classrooms for teaching basic computing. At the Foundation we use Scratch a lot in our Picademy courses, and it invariably involves physical computing (aka "learning computing while messing about and making cool stuff"). The Pi talks to the real world (and vice versa) via its GPIO pins, which can be connected to LEDs, motor drivers, sensors, goblin sticks and all sorts of other creative and wonderful gubbins. Of course the vanilla version of Scratch doesn't know about these pins. This made us sad—if you can't talk to the GPIO pins it limits the scope for physical computing. And if you can't do physical computing then the whole world is a sadder place.

So last summer we made a version of Scratch that talks to the GPIO pins without any additional software or fiddling. We call it… "Scratch." (It was a gruelling "name that software" meeting, with many biscuits and much tea.) It lets you do all of the usual stuff and more, from controlling LEDs and robots to using add-on boards such as Pimoroni's Explorer HAT and our own Sense HAT. You can even use the Pi camera to make instant sprite costumes. Even better, since September 2015 this version of Scratch is now included in Raspbian—the Pi's recommended operating system—so there's nothing extra to download, no other programs to run and no extra icons on the menu. It's a lovely thing.

We want to make physical computing as accessible, fun and creative as possible, and Scratch on the Pi helps do this. All of our resources using Scratch will use this version from now on and existing resources will be rewritten. A reference guide (bit.ly/1q0tfnF) shows how to use it with a beginners' tutorial. We'll be adding to the functionality and support for add-on boards as we go and we'd love to hear your thoughts and suggestions on our Scratch forum (bit.ly/25jJgFx).



The GPIO pins open up a whole world of physical computing

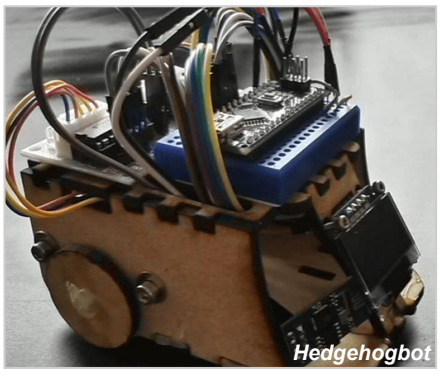
DIGIMAKERS FOSTERS CREATIVITY AND PROBLEM SOLVING

A group of Year 9 students from Colston's Girls' School enjoyed a day of electronics and coding at Digimakers, hosted by @Bristol Science Centre. Andrew Young reports on a packed day.

The day began with a textiles project using conductive thread and LEDs. With multiple workshops, wearing their illuminated baseball caps made it much easier to keep track of the girls' whereabouts during the day!

Later they had the opportunity to wear prototype devices from joint University research in soft robotics. The 'Sphere' project (irc-sphere.ac.uk/) aims to create technology promoting wellbeing in the home, funded by a £2 million grant from EPSRC. They had a number of ideas on show. A live data stream judged how smoothly a pancake was flipped. Detecting jerky movements in the elderly via a 'wearable' could be an early indicator of ill health. Soft robotic clothing could avoid falls, support whilst walking and give people added bionic strength to perform previously impossible movements. Ultimately soft robotics has the potential to replace stair lifts and other mobility aids. We hope to visit their 'smart' house, used to prototype new technologies next year.

Inspired by the European Space Agency ExoMars Mission in 2018 (bit.ly/1CYvBsd) a group were challenged to make an autonomous robot. The robot on show used infra red sensors on a rotating head to 'sweep' for obstacles left and right. Based on an Arduino, it was programmed using the Ardublock software, a version of the Arduino IDE that permits block-based programming. Learn more at bit.ly/1RWQKYc. The task really stretched our girls. Early attempts were anything but elegant, yet they persevered. They shared the delight of getting robots to explore their environment without bumping into obstacles; instead backing up elegantly and moving in the opposite direction.



Hedgehogbot

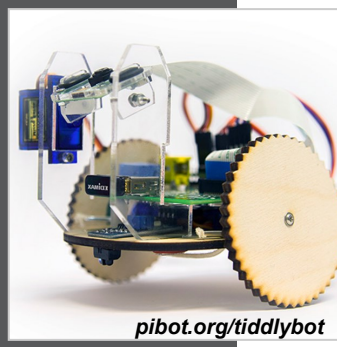
In a ciphers workshop one University student presented whilst the other coded in real time - a great way to show code taking shape. Another group of students hacked Minecraft using Python from a Raspberry Pi terminal. With plenty of experts amongst our students to get us up and running it made us consider a dedicated Minecraft server at Colston's!

A 'build your own smart watch' workshop also caught my eye with a group completely absorbed putting together and programming their own smart watch (above); a combination of a small Arduino, battery and screen.

What is compelling is the affordability of these electronic platforms. A uniting factor is physical computing - getting youngsters away from their screens, tinkering with electronics and robotics. They challenge the notion of what can be accessed in the world of Computer Science, ensuring participants leave with a wider view of the subject.

Digimakers run free, hands-on workshops for children (aged 7 to 18) using accessible, affordable and open source tools. More details and ideas at www.digimakers.co.uk.

Robots had a strong presence at Digimakers. They are a fantastic way to spark creativity and pique interest. UWE (University of West of England) were showing the growing family of Tiddlybot robots powered by Raspberry Pi. The project, from their world-famous Robotics Lab, has the potential to be a robust and polished platform for learning robotics and programming. Beautifully designed, it is increasingly feature-laden. Harry Gee, project manager, seemed to recognise that teachers want extensible systems that stand up to use and abuse. An evolving project well worth keeping tabs on.



pibot.org/tiddlybot

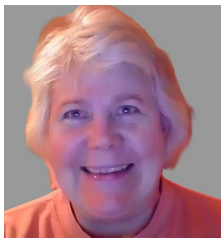
Hedgehogbot (bit.ly/1UU3fHS) was a great example of iterative design not only for participants but also the students developers. Digimakers harnesses their creativity, giving them a chance to develop projects into educational tools. Constant making and testing from enthusiastic participants quickly reveals their strengths and weaknesses. It has a laser-cut chassis (see lower left). The diagrams to connect the Arduino to the screen and motors were created with free software from fritzing.org. We hope to design a 3D-printed chassis and connect a wi-fi module to control the 'bot using a smart phone in our school Feb Lab. Gain an insight into building your own at bit.ly/1V7AV48.

CAMJAM RASPBERRY PI WARS

On Saturday 5th December , three year 13 students from Lutterworth College took part in the second national Raspberry "PiWars" contest at Cambridge University, along with three members of staff. Over the previous 10 weeks these students, along with others from Years 11 and13 attended the after school Coding Club to help develop a part autonomous/part manual robotic vehicle which would be able to compete in a number of challenges such as 'Pi Noon', straight line speed test, skittles and aesthetics.

Lutterworth's robot was developed using mostly secondhand, donated equipment, but was competing against some teams that had spent a considerable amount of money on design and development. This did not deter the students and they made the most of the opportunity to talk to other competitors, who particularly liked our dog's head with its flapping ears! Although we did not win any of the categories outright, the robot was able to compete to an acceptable mid-table-ranking in most of the challenges. One of the highlights was the team being presented with a RaspberryPi Zero by Eben Upton – one of the founders of the RaspberryPi Foundation. Maybe it's time to think about entering the 'Smallest Robot' category next year? *Andrea Keightley*

MICROSOFT'S ANNUAL THINK COMPUTER SCIENCE



Fakenham High School in Norfolk have attended 'Think Computer Science' for the last five years. Aimed at KS4 students, Sue Gray explains why it is so valued.

The one day event gives students a flavour of the types of roles that come under the broad umbrella of 'computer science'. This year's began with two very interesting talks, the first from Steve Hodges, Research Manager/Principal Researcher at Microsoft Research. He told us about his personal journey to his current role and, more importantly for students, showed them there are many ways into computer science – and programming is not necessarily the most important! Philip Bielby, Technical Director at Runescape followed – I could see some of the students' ears prick up! Clearly they recognised the brand. Philip illustrated the varied roles in game creation, considering the ways in which people play games now and in the future.

Off for our first visit to the Science Fair. There was cs4fn showing magic tricks which captivated students and staff alike. Students could play with a Kinect and 'see' their own hand rendered and animated as if for a game or other virtual environment. The queue at the Runescape stand was to try the virtual reality games on show and the stand handing out VISR gadgets soon ran out of supplies. Students said they really enjoyed playing the 'old school' games at the Centre for Computing History whilst right up to date next door was Kodu coding.

Two workshops both focused on the micro:bit this year and students quickly got to grips with the Touch Develop environment. By the end of their sessions they had all made animations, created scrolling text, learned how to use the on-board accelerometer and created a 'sorting hat' program. Being Hour of Code week it was a great way to spend time.

Mark Calleja does some amazing workshops in Cambridge teaching how to program but in such creative ways youngsters barely realise they are doing it. He explained to students how they could be 'white hat' hackers, insisting the only limit to creativity was their imagination.

Woven through the day was the Forza Challenge. Our own Max was clocked up a superb time of 1:17, retaining his lead throughout the day before taking part in the final head-to-head race. A real cool customer, he didn't let the venue or the pressure of the event put him off. He won convincingly and took away a Minecraft Gameband plus £1000 for the school (see the photo top left)! The whole group were buzzing and jumping around and smiling like crazy – what a fabulous day.

'Think Computer Science' (TCS) is incredibly stimulating, challenging and fun. Student learn about the possibilities of computing and where it can lead as a career. They also get to experience some cool 'toys' and gadgets, some of which are not yet available to the general public so there is a real sense of being 'in on the ground floor'. I love TCS and my students do too. They always ask, "Can we go again?" And their smiles will tell you all you need to know – a brilliant day out and so very interesting and educational.

EXPLORING THE LIMITS OF WHAT COMPUTERS CAN DO

There is a lot more to computation than first meets the eye. John Stout, from King George V College, Southport recommends Computers Limited by David Harel.

This book is on the limits of computers, but to do that the author first shows what they **can** do and, to a certain extent, how they do it. It's written for the non-technical audience but is still quite a challenging read, covering hard computer science topics but in an approachable way.

What's it all about covers the basics of algorithms by looking at problems such as making a chocolate mousse, multiplying two integers, and calculating a company's salary bill, before posing 8 short problems, which, by the time you've finished, you'll be able to put into different categories.

Sometimes we can't do it? starts to introduce the limitations of computers by introducing problems that are *undecidable* starting with the *tiling problem* (see bit.ly/1o4nBzo for an excellent introduction from cs4fn). This is one of the problems that we cannot write a program to solve, no matter how powerful our computer or language. Here Harel introduces the Turing machine, a model of computing that can solve any computable problem. The news gets worse (or more interesting) as you read further. A variation of the tiling problem turns out to be decidable on an infinite plane but undecidable if you are restricted **in any way**. Turing's *halting problem* (**HP**) makes an appearance and it turns out we can't write programs to tell us **anything** (non-trivial) about **any** program. Problems are connected. If we had a program which solved the halting problem (which we can't) we could solve the tiling problem too.

Just when you think it can't get much worse the concept of *less decidable* problems arrives. Can we write a *program verifier* (**PV**) that takes a program **p** and a problem description **d** and tells us if **p** solves **d**? It's not surprising that the **PV** problem is undecidable but what's curious is that if we did have **PV** we could use it to solve **HP** but not vice versa, so **PV** is *less decidable* than **HP** and Harel's initial division of problems into just **computable** and **non-computable** turns out to have a third category: **highly non-computable**!

Sometimes we can't afford to do it looks at searching and sorting, and the Tower of Hanoi problem via an iterative algorithm. A program that can handle inputs of 10 in under a second takes many centuries to handle inputs of just 200! Here the concept of **tractability** appears. Harel adds **tractable** and **intractractable** to his problem categories, though it's no surprise that some problems are **highly intractable**.

Sometimes we just don't know introduces another tiling problem (the *monkey puzzle*) which, for a reasonably sized input takes over 500 trillion trillion years! This introduces the **NP** and **NP-complete** problem classes and the concept of a magical coin to choose the 'correct' path to a solution. With a non-deterministic coin (or a magical processor instruction) **NP(-complete)** problems become **tractable**. The search to find that coin remains one of the great challenges of computing.

TRYING TO EASE THE PAIN

The final chapters introduce new topics (*parallelism, randomization, quantum computing* and *molecular computing*) which might help solve problems more quickly. *Parallelism* (throwing multiple cores at a problem) is considered overleaf. A *Monte Carlo* algorithm uses randomness to produce a quick but possibly incorrect answer. A randomized algorithm could test large numbers for primality quickly and be as certain as required, e.g. with only a 1 in 2^{500} chance it is incorrect. Watch Simon Peyton-Jones explanation on CAS TV for a better explanation!

Quantum computing uses the 'weird' properties of quantum particles (*superposition*, *entanglement*, and *interference*) so that, in theory at least, we can achieve an exponential speedup in solving problems such as factoring a large number. By using *qubits* rather than traditional *bits* a system represents all the possible answers to a problem. All that needs doing is 'arranging things' so the correct one 'drops out'. Of course, it's the 'arranging things' that is currently receiving, and needs, a lot of attention. *Molecular computing* uses billons or trillions of molecules to solve a problem, depending on biological properties to produce the correct answer to a problem.

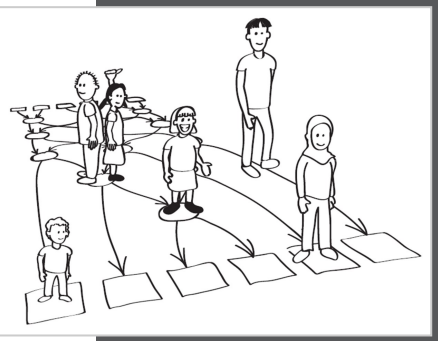
Turning bad into good turns difficult problems to our advantage, factoring numbers in cryptography being the main example. Finally *Can we ourselves do better?* looks at Artificial Intelligence, chess playing and the Turing Test but is best for its introduction to *heuristics,* knowledge and understanding natural language. You can't fail to be impressed by the breadth and depth of this book.

Moore's Law as mentioned below originated with Gordon Moore, the co-founder of Intel. Curiously, in 1952 Moore gave one of the first descriptions of how to make a physical Turing machine (TM). This was a purely academic exercise: TMs are a very poor basis for real computing compared with Von Neumann machines, as Moore undoubtedly knew. We will return to Turing machines in a future column.

INTRODUCING PARALLELISM IN THE UNPLUGGED CLASSROOM

Many children will find Moore's Law doesn't accord with their experience. For those coming to Computing in the last decade there is little evidence of the former explosive growth in transistor density, and associated hardware developments. However, it is easy for them to appreciate they may be witnessing a new exponential development. The introduction of multicore processors has happened in their lifetime and helps give a sense of historical place. Are they in at the start of an age that may come to be characterised by massively parallel systems? It is an exciting prospect.

Accomplishing tasks in parallel presents new challenges in scheduling, both for Cloud based applications and those running locally on multi-core computers. A wonderful unplugged activity can help get the idea of parallel processing across in a simple way.



The National Curriculum at KS3 asks that students consider well known algorithms. The Network Sorting activity from CS Unplugged, allows students to reflect on how comparisons made in parallel can impact on the effectiveness of a sorting algorithm. Take a look at youtu.be/30WcPnvfiKE to get a sense of how much fun this activity can be. Resources to support the activity can be downloaded from csunplugged.org. A follow up homework might be to read a child friendly introduction to the idea in two short articles from CS4Fn: bit.ly/1Vc8z7T and bit.ly/1MX1KQV. *Roger Davies*

WHY SOMETIMES IT'S QUICKER JUST TO DO IT MYSELF

Greg Michaelson, Professor of Computer Science at Herriot-Watt University in Edinburgh considers both the practical potential and limits of parallel processing.

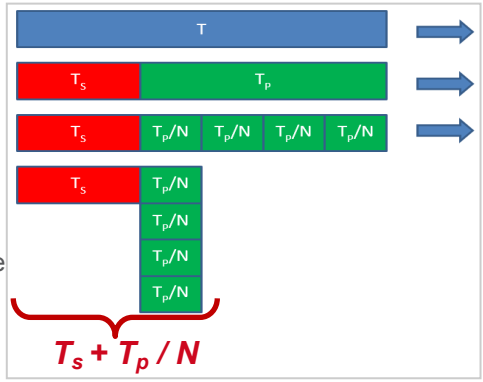
Moore's Law suggests that, subject to ultimate physical limits, the number of transistors that can be realised on a given area of silicon will double every year or so. This was formulated in 1965 by Gordon Moore. Until quite recently, the practical implication of Moore's Law was that CPUs went on getting beefier, by increasing the data and instruction width, from 4 to 8 to 16 to 32 to 64 bits, as well as by upping the clock speed. But, at the start of the 21st century, general use processors seem to have hit a 64 bit/3GHz ceiling. These days, instead of faster CPUs, we get more of them. Thus, two and four core CPUs are common, even on mobile phones, and processors like Intel's Xeon Phi offer fifty or more 1GHz cores at surprisingly low prices. The strange thing though is that despite all these additional cores, our devices don't seem to get much faster. I think that there are a number of reasons for this.

First of all, there are physiological limits to humans. In particular, it takes us around 200 milliseconds on average to respond to a visual change, quite apart from the limit to how often we can click a mouse or keyboard. So, even if our software is faster, our speed of interaction with it doesn't change.

Secondly, adding more cores won't speed up software that can't take advantage of them, and most extant user software was written for one CPU. Contemporary operating system will try to schedule separate processes to run in parallel on different cores, and that will make a difference if we try to run lots of programs at once. But once there are as many cores as active processes, adding more cores won't make any difference as every process effectively has sole use of one core.

Most important, even if we try to craft software to take advantage of multiple cores, say by splitting our program into multiple threads, we'll be lucky to achieve anything approaching a uniform improvement. The reason is that *there is always part of a program which is inherently sequential*, for example input/output, especially file and network access, and program initialisation and shut down.

Ideally we would like to achieve *linear speedup*, where a problem that runs for time T on one core takes time T/N on N cores. Suppose the total time is $T = T_s+T_p$, where T_s is the time for the part of the program that must be sequential, and T_p is the time for the part that may be run in parallel.



Suppose next that we can perfectly parallelise the part that takes time T_p, so, with N cores, each core will take time T_p/N. Then our new overall time is T_s+T_p/N, which is always greater than T/N:

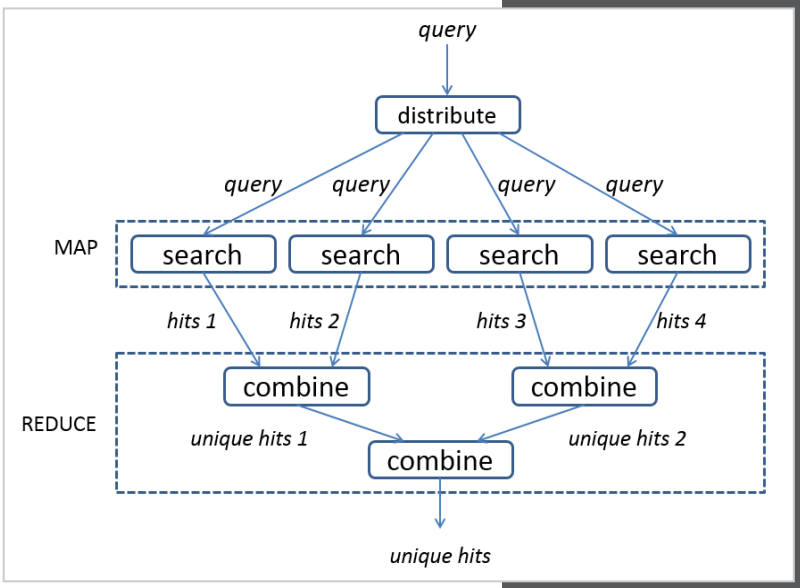
sequential time T_s	parallel time T_p	number of cores N	total time T_s+T_p/N	speedup $(T_s+T_p)/(T_s+T_p/N)$
10	90	1	10+90/1 = 100.00	1.00
10	90	10	10+90/10 = 19.00	5.26
10	90	100	10+90/100 = 10.90	9.17
10	90	1000	10+90/1000 = 10.09	9.91

We can measure *absolute speedup* as the ratio of the time for the original sequential program on 1 core to that of the parallelised program on **N** cores: $(T_s+T_p)/(T_s+T_p/N)$

For example, suppose that, for some program, the sequential time on 1 core is 100 units, I/O takes 10% of the time and we can perfectly parallelise the other 90%. Then, with 1, 10, 100 and 1000 cores we get speedup as outlined in the table at the top of the page. With 10 cores, our parallelised program goes a bit more than five times faster than with 1 core. But with 100 cores, our program doesn't quite go ten times faster, and with 1000 cores it is hardly faster than with 100 cores. *Adding cores cannot affect the sequential time* T_s, which comes to dominate the overall run-time.

Of course, in practice we can't perfectly parallelise the potentially non-sequential program part. Each parallel component will have an additional overhead to communicate with the top level program, to receive data and send back results. This fundamental limitation to parallelisation is known as *Amdahl's Law*, and was first formulated in 1967 by Gene Amdahl, the systems architect for the IBM 360 range. These mighty mainframes dominated international computing in the 1960s and 70s. IBM still makes mainframes, but these days they're essentially servers full of multi-core processors, not just a single CPU like the 360.

We can find lots of real life analogies to Amdahl's Law. For example a well worn parody maths question asks us to work out how long it will take *N* people to dig a hole if it takes 1 person *T* hours. The answer is supposedly *T*/*N*, on the assumption that each additional digger has the same capabilities as the others and so will reduce the total time by a uniform amount. Alas, for real holes and real people, adding more diggers can have less impact than we might like: the diggers get in each others' way, as well as spending time talking to each other instead of digging. Thus, for a given hole, there will tend to be an optimum number of diggers.



Similarly, a common drawback when someone offers to help us with something is the additional time it's going to take us to explain to them what to do, and then to guide them as they're doing it. Hence the old adage: "It'd be quicker to do it myself."

PARALLELISM IN PRACTICE

Google's ***map-reduce*** framework is used to build scalable parallel systems on vast computer farms to search enormous quantities of web data. In turn, map-reduce is explicitly based on ideas from functional programming. The map higher order function applies some other function to every element of a sequence to return a sequence of results. For example, we might map the "times 2" function across an array of integers to double each element. If the treatment of each element does not depend on the other elements, as with doubling, then the function could be applied in parallel to all the elements.

In contrast, the reduce higher order function applies a function across a sequence to combine the elements. For example, we might find the sum of an integer array by adding successive elements to a running total. If, as with summing, the order of reducing the sequence is not important, then the process can be realised as a parallel divide and conquer.

So, for a web search, Google first deploys map to inspect large numbers of data sets in parallel. Many of these separate searches will result in the same hits, so Google then deploys reduce to combine them into a list of unique hits to show to the user.

The beauty of map-reduce is that the map and reduce components can be instantiated with *any* functions, not just for searching and combining. Google's map-reduce is proprietary but Apache offers the open-source Hadoop. This is written in Java and used, amongst others, by Yahoo. More details of Google's map-reduce at bit.ly/15V7Pyh and Hadoop, bit.ly/1MkD2yR.

A PAUSE FOR THOUGHT

In 1997 the chess game between Gary Kasparov and 'Deep Blue', from IBM came to symbolise the advance of Artificial Intelligence. Hype and extravagant claims followed, and, not for the first time, when wild aspiration failed to be satisfied, AI slipped swiftly out of the public eye. The symbolic breakthrough, the hype and subsequent disappointment seem to characterise the history of AI.

The allure (and fear) of a 'thinking machine' is as old as antiquity. Recognising the potential of the Analytical Engine, Ada Lovelace reassured the public that it could do no more than it was programmed to do. Alan Turing gave birth to the modern notion of AI. His ideas germinated in the heady environment of early electronic computers. The famous Dartmouth Conference in 1956 with contributors such as John McCarthy, Marvin Minsky and Claude Shannon generated a huge sense of excitement; grandiose claims and, with hindsight, much wishful thinking. "Within ten years a digital computer will be the world's chess champion" was Alan Newell's bold prediction. The long 40 year wait for the first chess victory dampened much initial ardour.

Yet if we consider other claims made for AI at that time, in the light of recent advances in machine learning, the forward march of the discipline is remarkable. Compared to chess, the rules of Go may seem simplistic, but they mask the strategic complexity of the game. When AlphaGo beat champion Lee Sedol 4-1 recently, a new frontier was breached. Whilst the press response has been muted, could this be a new dawn for AI? And shouldn't we be exploring ways to teach our children about the ideas involved?

COMPUTER SCIENCE, CS4FN AND SOME WILD SOUNDS

We tend to think of computing as being all about technology, and so not for people interested in subjects like biology or ecology, which are about people, animals, plants: the study of life. However, computer science is, above all, an interdisciplinary subject, and many computer scientists study living things too.

Issue 21 of QMUL's Computer Science for Fun (cs4fn) magazine, which will be sent free to subscribing schools in the summer term, is all about sounds in the wild. It is funded by an EPSRC research project concerned with developing programs that can identify bird song. The magazine will explore different ways to solve this and similar problems: should you engineer rules that identify each bird, or let the computer learn for itself what sound is a robin and what is a blackbird (and what for that matter is a plane overhead or a human pretending to be a bird)? It will also look at why this is important - not just for bird watchers, but for scientists studying climate change, and biodiversity too. Computational thinking skills are useful wherever your interests lie.

STEVE FURBER DISCUSSES COMPUTERS AND BRAINS

Steve Furber discusses his research in the latest addition to CAS TV. One of the designers of the BBC Micro hardware and the ARM chip he is now ICL Professor of Computer Engineering at The University of Manchester. Steve uses 500,000 (yes, half a million!) ARM cores in SpiNNaker, a massively parallel computer designed to model a portion of the brain. Not to be missed. Make sure you receive notification of the ever expanding content on our new CAS TV channel. Subscribe today: youtube.com/computingatschool

CAS TV

COMPUTING AT SCHOOL

EDUCATE · ENGAGE · ENCOURAGE

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

Many thanks to the following for help and information in this issue: *James Abela, Clive Beale, Irene Bell, Tim Bell, Miles Berry, Paul Curzon, Claire Davenport, Roger Davies, Stuart Davison, Lorna Elkes, Catherine Elliott, Sue Gray, David Hill, Lyndsay Hope, Pete Jeffreys, Andrea Keightley, Catriona Lambeth, Emma-Ashley Liles, Duncan Maidens, Simon Marsden, Adam Martin, Greg Michaelson, Emma Norton, Ben Nuttall, Michael O'Kane, Simon Peyton-Jones, Zoe Ross, Marc Scott, Andrew Shields, John Stout, Mark Thomber, Carl Turland and Andrew Young.*

www.computingatschool.org.uk

Computing At School are supported and endorsed by:

bcs The Chartered Institute for IT
Making IT good for society

Microsoft Research

Google

CPHC
The Council of Professors and Heads of Computing