COMPUTING AT SCHOOL
EDUCATE · ENGAGE · ENCOURAGE

SWITCHEDON

COMPUTING AT SCHOOL NEWSLETTER WINTER / SPRING 2012

CELEBRATING THE GENIUS OF ALAN TURING

“We in Britain should never forget that one of our great heroes, Alan Turing, laid the foundation stones on which all modern computing rests.” In his BETT speech (see right) Michael Gove highlighted 2012 as marking the centenary of his birth. Some students may be aware of the name, largely because of his code breaking activities during WW2. Some may know about the tragic nature of his death; a persecution that is hard for many to grasp today, though it happened little more than fifty years ago.

But how many will be able to recount any details of his key ideas? In the preface to his excellent book, ‘The New Turing Omnibus’, AK Dewdney pays tribute to “...Alan Turing, whose ripe mathematical imagination was fuelled by fantasies of mechanized intellect. Turing’s greatest single legacy, the Turing machine itself, became the best known vehicle for the emerging theory of computability. Wonderfully simple, with only two moving parts, it can be explained to the man or woman in the street, yet it embraces everything we mean by the word ‘computable’!”

Turing was a genius ahead of his time, yet few pupils are aware of his place in history. Computing has been hidden from the curriculum for far too long. Many wrongly equate computing with just learning about programming. It’s far deeper than that. You’ll find pointers to Turing’s ideas inside this newsletter and more in the online supplement. The centenary is a marvellous opportunity for teachers to resurrect some key computing concepts; concepts developed by Turing before an electronic computer had even been built. We’ll cover some resources in each issue this year. A wonderful five minute BCS video, made as part of their Pioneers series, is the perfect starting point. Why not use it for an assembly?

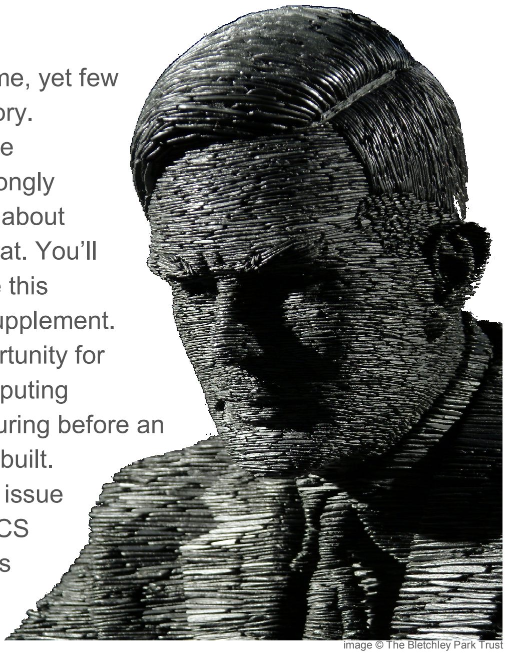


image © The Bletchley Park Trust

GOVE’S SUPPORT FOR COMPUTER SCIENCE

"Computer Science is a rigorous, fascinating and intellectually challenging subject”. So says Michael Gove, Minister of Education. He continued his keynote speech at BETT by saying; “Although individual technologies change day by day, they are underpinned by foundational concepts and principles that have endured for decades. Long after today's pupils leave school and enter the workplace - long after the technologies they used at school are obsolete - the principles learnt in Computer Science will still hold true". This is the clearest statement yet from the Government.

Two days later, The Royal Society published its 18 month study, “Shutdown or Restart” carefully identifying three separate strands that comprise Computing in schools. They make 11 recommendations to take the subject forward.

Michael Gove cited the CAS Curriculum as one source schools should look to when planning new curricula. CAS look forward to supporting teachers as they plan for September and beyond. We also look forward to working with others in creating a curriculum that inspires our youngsters with the magic and beauty of computer science.

BASIC
>

THE BLINKING CURSOR

The 80s were rubbish. We were promised Bacofoil jumpsuits and teleporting to school. Instead we got Cabbage Patch Kids and strikes. Fortunately we also had the best thing ever invented: low cost home computers that plugged into your telly. The mighty BBC Micro; the po'boy ZX Spectrum; the painfully hip Commodore 64; plus a lesser pantheon of Dragons, Orics, Lynxes and more.

Playground fights over which machine was best were not uncommon. Yet these computers all had something in common: when you switched them on they just sat there, blinking. Waiting for you to tell them what to do. This was a wonderful thing. The micros came with programming manuals and there was an irresistible urge to make them do stuff. So we hacked and we POKED and we coded and created. At least until Mum kicked us off the telly because That's Life was on.

The blinking cursor single handedly made the UK tech and digital creative industries the envy of the world. Never mind that Google's Eric Schmidt recently found our "great computing heritage" stuffed in a skip round the back of Poundland. The 8-bit revolution is what kicked it all off.

The Raspberry Pi Foundation wants to reboot this revolution. It wants to recapture the fun and excitement and creativity of these early micros; to get kids coding again and to foster a new generation of computational thinkers and digital creators. Its plan? To make a low cost home computer that plugs into your telly. That sits there and blinks at you.......

REDISCOVERING THE SPIRIT OF THE BBC MICRO: RASPBERRY PI

Even before its launch, the innovative Raspberry Pi is already gaining widespread acclaim. But how did it come about? In an exclusive interview Clive Beale talks to Liz Upton about what spurred its development and motivated the trustees.

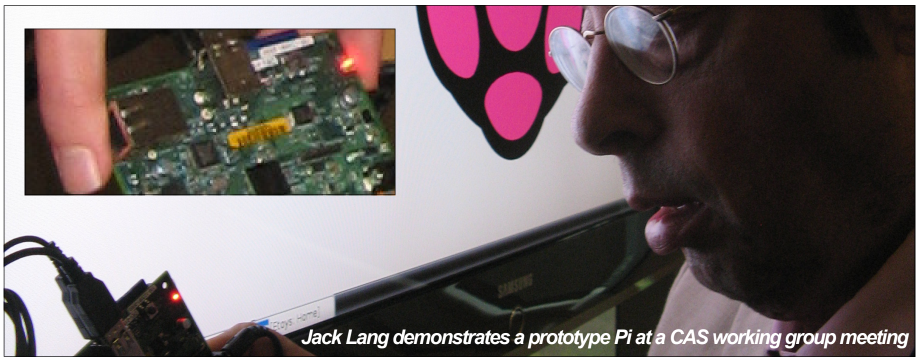
Where did it all start? Was there a light bulb moment?
There wasn't a eureka moment; more a growing, horrible realisation that things in school computing were not as they should be. Eben (Upton) was working at St John's College, interviewing kids applying to the university. There was a noticeable drop-off in the quality of applicants. In the 1990s, those coming to Cambridge would arrive equipped with a few programming languages and several years' experience of hobbyist hacking. By the 2000s, things had changed significantly. The typical applicant had perhaps written HTML, used CSS or created a macro or two, but that was usually the extent of their 'programming experience'. The numbers applying were falling too.

We talked about the causes. A number of factors are in play, but one big culprit seemed to be the disappearance of computers which are easy for kids to program. These vanished with the ubiquity of the Windows desktop. We grew up with BBC Bs, Amigas and ZX Spectrums. You had to type something to make them do anything. My Beeb, and the Beebs at school, booted into Basic; every kid in school knew how to walk into WHSmiths and run a little program making all the display machines say "This shop smells of cheese." (We also knew how to sprint out of WHSmiths. They called it a rounded education!) Programming was much more prevalent in school; the curriculum has changed to an unrecognisable degree.

I told a neighbour's kid about Raspberry Pi. His ears pricked up; he told me how he'd love to learn to program, to work in games – but his family don't have a computer. Mum's a cleaner, Dad drives a van. Neither comes into contact with computers in their daily lives and they're not a necessity in the house. They have a Wii, which the kids use to browse the internet, but a Wii is a closed box that no kid is going to be able to program. It's a story we're hearing over and over again. Twenty per cent of UK homes don't have a computer. Since schools stopped teaching programming such kids have no way into computing.

After an evening complaining, Eben got out some breadboard, ordered an Atmel chip and a pile of transistors, and sat down to solder together a little computer that booted into Python, with an idea to build a kit that perhaps you could sell to schools. I got to hold the voltmeter. Five years on, one room has been converted into an electronics lab, and that first board and Broadcom chips have had babies which turned into Raspberry Pi.

If I was gathering a team to take over the computing world, the RasPi Trustees is where I would start. Apple started with a lot less :). How did you get together?
Eben brought the team together from friends who work at the Cambridge Science Park (David Braben); old colleagues from the university (Jack Lang, Alan Mycroft and Rob Mullins) and friends of theirs (Pete Lomas). They were picked for their experience – David is, quite simply, legendary in British computing; Jack worked on software for the original BBC Micro and teaches business; Pete runs a PCB fab that's one of those British technological success stories; Alan and Rob's educational contacts and experience are as good as you could


Jack Lang demonstrates a prototype Pi at a CAS working group meeting

hope to find. They're a really strong team, and I think we'd have struggled to put together a group like this anywhere outside Silicon Fen.

In a project like this, where we rely so heavily on the open-source community, transparency and a genuine gusto for what you're doing is essential. I'm preternaturally enthusiastic about the project. I've been living with generations of circuit boards stacked all over my house for years, so I was a good fit for getting the Pi community up and running.

And the name? Is it a riff on Apple? Is there a nod to Python?
It's partly an in-joke. Fruit names were a nice tradition in computing (Apple, Apricot, Blackberry, Tangerine and all that jazz). I think Rob suggested the raspberry, which is a friendly fruit – Pi, a reference to Python, came from Jack, an ex-restaurateur and fan of pastry.

The original charitable objective is: "to further the advancement of education ... particularly in the field of computers, computer science and related subjects" Has this changed in the last two years?
When the community started to respond to the idea we realised the potential scope was far, far wider. After David appeared on Click Online (BBC, May 2011), we started getting thousands of mails a week. Many are from groups priced out of traditional computing. Suddenly, the impact of what we're working on started to look…a little scary, if I'm totally honest. We have to get this right.

Do you see it as a game changer?
Lord, we hope so. We're not really innovators in size or compute power. There are other machines out there that do a lot of what Raspberry Pi does. But we are innovators on cost. Success would be competitors driving costs down so they're accessible to everybody. The fact that it's possible to produce hardware at this sort of price, running open-source software, exposes how expensive business software is. When a license is ten times the cost of the hardware you're running it on, that's quite startling.

Broadcom are instrumental in the project. Is this an experiment, or a favour? Do they share the Foundation's goals?
Broadcom sell us chips on commercial terms; they aren't subsidising us, as some people believe. They're doing us favours in agreeing to sell us chips at all, because our volumes are millions short of what they usually sell in; and in allowing some engineers to work on the project in their spare time. They believe in our goals; we've had support and reassurance right up to CEO level. On a purely business level, ensuring graduates can program is a good thing. They have a very active charitable arm (The Broadcom Foundation), much more responsible than some of the more opinionated open source community would have you believe! That has hooked us up with charities, especially in India, which we hope can push the device into the developing world.

Jack Lang said, "there's no point in inventing a better mousetrap if mice aren't a problem." Have you understood the problem? Is Raspberry Pi the solution?
If we reach one kid per school who has a latent talent we'll have done what we set out to do; reinvigorating the demographic applying to university and later, jobs in UK tech companies. We can do much more than that, but the proof of the pudding is in the eating. We would love to be able to bring more girls into the subject, but I'm not sure that that's something the Foundation can address; it needs to be happening in schools, at a classroom level.

Many people want to change the world. Most don't. Why is Raspberry Pi different?
It's a mindset thing. I don't subscribe to the notion that one person can't make a difference. You've got to be prepared to work hard if you want to change things – but if you're working on something you're passionate about, it's not really work, just a very supercharged hobby that requires a serious caffeine habit. *Clive Beale*

RASPBERRY PI FOR EDUCATION USER GUIDE

An education version of the Pi is planned for release this spring. It was very clear to CAS that a user guide was a must, and with the Raspberry Pi team already overstretched we decided to write one ourselves. The guide will teach two languages, Scratch and Python, through creative scenarios such as music and graphics. It will also include advice and resources for budding computer scientists. The outline of the guide is already written and there is currently a huge amount of work going on backstage to develop the content. The guide will be available as a free download and we are investigating the possibility of a printed version. Paradigm shifts are ten a penny nowadays. Everything from smart phones to air fresheners are described as "game changers". But in a couple of years we may well look back and see that the Raspberry Pi deserves this sticker. We hope that the CASpberry Pi user guide will contribute to its success as a learning tool and help inspire a new generation of coders.

SCHOOL PROTOTYPING PILOT WITH NEW .NET GADGETEER

Microsoft Research have recently released an exciting new product to enable the prototyping of a huge range of gadgets. Sue Sentance reports on a school pilot that has been running in eight schools in Essex and Cambridgeshire.

.NET Gadgeteer was developed by Nic Villar and James Scott. The first kits are now available for sale from GHI Electronics. It has great potential in schools as it can be used to teach students simple electronics, computer programming and computer-aided design. A digital camera can be built in about half an hour!

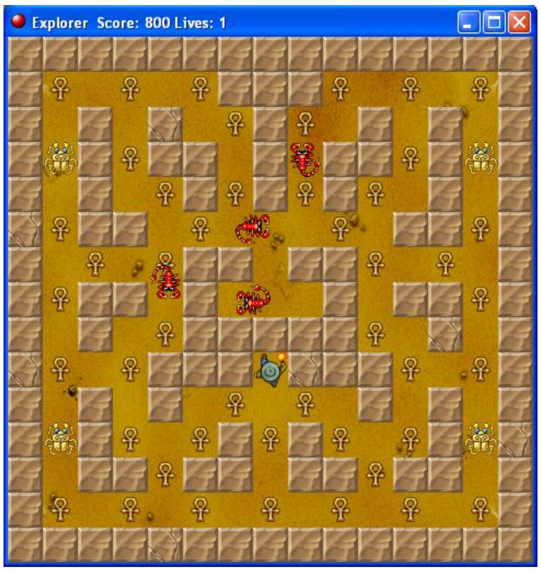
A school pilot has been running in Essex and Cambridgeshire to find out how secondary school children get on with the programming of the Gadgeteer. Eight secondary schools are involved. After-school clubs were planned for Year 9s to try out the teaching materials that I am developing. The pilot was launched at a workshop on October 6th. The teachers spent the afternoon building gadgets using .NET Gadgeteer and had the opportunity to talk to the developers about tricky bits of programming. The pilot will end with a grand "Show and Tell" at Microsoft Research on January 30th 2012 where all participating children will be able to demonstrate the gadgets they have developed themselves with the best being awarded prizes.

The after-school clubs are well underway. There have been no shortage of children wanting to take part. Niki Smith, a participating teacher says of her club "the children are having a lovely time with Gadgeteer and so are the teachers. They talk about it during the week. They are not put off at the first hurdle when the code doesn't work, they are quite happy to fiddle around and find their way through it."

Sue Sentance

GAMEMAKER: A GREAT WAY TO LEARN HOW TO PROGRAM

Graham Hastings was shown GameMaker by a pupil in his computer club nine years ago. They created professional looking 'arcade' games within a few hours. He explains the appeal and why he is still using it today.



An example of a simple GameMaker arcade style treasure game

At that time I ran a small club for children (aged 11 to 13) who were interested in developing their computer skills. Using GameMaker it is possible for children as young as 7 to program events using simple, easy-to-learn, drag-and-drop actions. They amaze themselves and their friends when they successfully create professional-looking games. Once the children are more experienced, I encourage them to use the built-in GameMaker Language (GML). They can work with existing 'blocks' of code or write their own scripts from scratch. Using GML it is possible to teach all the programming constructs children might learn through working with a mainstream programming language.

Until I saw GameMaker I tended to teach Visual Basic because the children liked to create their own little Windows style, self-contained, applications. They weren't keen on the rigour of testing though. With GameMaker, the children are eager to test their creations on their friends and take the comments made by their peers seriously. When developing games the performance of their application is a major concern It is particularly rewarding to see how eager they are to amend their games in light of the advice received. The children display a competitive edge. They want their game to be the 'coolest' and become involved in a race to develop them. They are, in short, fully engaged with the design process.

Teaching GameMaker requires little prior knowledge. There are oodles of tutorials and lots of guides posted on YouTube (many by children). YoYo Games has developed an extensive, active and supportive community. It serves as a space for children to seek help and download games and tutorials – everything a teacher could wish for. Moreover it teaches them to learn independently. Many of my pupils are much better GML programmers than I am and know not to come to me expecting answers to their problems – they have other strategies which include posting questions and deconstructing existing games. Is that not the modus operandi of all commercial programmers? And what of the club? Well it is still going strong but it is no longer small, nor is restricted to the 11 – 13 age range. We now have a club for 7 and 8 year olds.

Graham Hastings

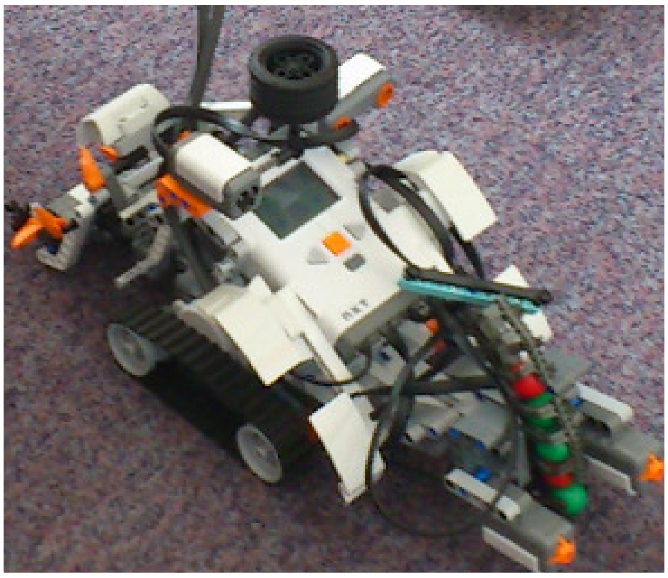
INTRODUCING ROBOTICS: HAVING LOTS OF FUN WITH BITS AND BOTS

Lego Mindstorms NXT kit is proving a hit in a lunchtime club in Monmouth. Its easy interface has hooked the students, but, as Lyndsay Hope explains, the real benefit lies in provoking them to explore further into the capabilities of 'real' languages.

Last year we started working with robots in a Year 9 lunch time club. The students worked in small groups of three or four, equipped with the NXT kit and a recipe for a basic 'rotate and shoot' sentry bot. It guided them through the process of building their bot and creating basic programs using the built in software with its drag and drop interface - very useful for those completely new to programming. The students were shown how to upload a program, operate and test its movement, introducing ideas like sequencing, loops and iteration.

Initially the computer was used to control the robot using its built in Bluetooth connection, however we discovered a free Android app called MINDdroid, that did the job just as neatly from a mobile phone. This gave the freedom to walk the bots about, exploring different terrain and new challenges. It also introduced greater ease of control, making handling more fluent, much like a game controller.

Having built the initial bot from the given instruction set, students were very keen to create different models.



We ran a Robot Challenge, over a range of events including a Maze, Sprint, Shot-Put and Target Shooting. The students designed and built their bots, testing them and reworking their designs as needed. Their particular aptitudes came to the fore in the process, some focussing on design elements, whilst others looked at refining the programming for manoeuvrability, improved sensing and more accurate shooting. Most of those Year 9 students have returned to the club in Year 10 with new ideas and a keenness to explore still further.

We have also embarked on an a multi-school science project for Sixth Form students who have never programmed but are keen to explore the possibilities. The learning curve is shallow and students are excited by the swift results they get; they are keen on the creative aspects and on the opportunities they have to develop and experiment, to ask "What if?" and get immediate feedback.

Looking ahead, the next project involves programming the robots using Microsoft Robotics software (see supplement) moving away from the visual, drag and drop interface to coding and giving other choices, for example using XBox controllers to control the bots. This is proving interesting for senior students and also provides some of our juniors with useful examples of practical coding in a very engaging way.

Lyndsay Hope

Animation12

The submission deadline to the annual UK schools animation competition is March 23rd. Promising to be even bigger than previous years, pupils between the ages of 7 and 19 are challenged to produce a one minute key frame animation. Initiated by the University of Manchester five years ago, Animation 12 has a huge following in schools. Last year 563 schools registered, with 886 entries received. Some 45 students from 25 schools won prizes last year.

The galleries on the Animation 12 website will provide inspiration. Further details about how to register are in the supplement. A greater range of software is permissible this year, including Scratch, Alice, Flash, DrawPlus, KoolMoves, SWiSH Max4 and Blender. Animations can be based on any theme but this year there will be a special category for animations based around the life, work, ideas and impact of Alan Turing. Turing worked with the University of Manchester following WW2, where computer pioneers developed the first stored program computer, 'The Manchester Baby' in 1948. A separate competition, Codebreaker, using Greenfoot, has been organised in conjunction with CAS. See overleaf for details.

VPL CODE FOR ROBOTICS

Microsoft's Visual Programming Language (VPL) for robotics is aimed at novices, based on a "graphical dataflow-based model. A dataflow program is more like a series of workers on an assembly line, who do their assigned task as the materials arrive. As a result VPL is well suited to programming a variety of concurrent processing scenarios." See the web supplement for further details of how to access this free resource.

CODE BREAKER

To celebrate the centenary of Alan Turing's birthday in 2012, the Computing At School group has launched "Codebreaker", a Greenfoot-based programming competition for school students. A full range of prizes is available for each age group and the competition is free to enter.

CAS are delighted that the Codebreaker competition is being offered as part of the very successful Animation12 run by the University of Manchester and enables students to provide interactive programmed solutions on the theme.

In launching the competition CAS co-ordinator Simon Humphreys said "Alan Turing was such an important figure in the history of computing; a national competition aimed at encouraging students to design and implement a program on the codebreaking theme is a fitting way to celebrate Turing's life and work at Bletchley Park".

Using the Greenfoot IDE, students have to create an interactive program which is linked to theme of "Codebreaker", either as a game or a puzzle. It is not intended that the player task is to necessarily break a code - a general link to the theme is sufficient.

The CAS website has a suite of resources to support teachers including a 10-week after school club programme culminating in work devoted to the Codebreaker challenge plus supplementary resources for researching the life and contribution of Alan Turing and cryptography. For links see the web supplement.

Entries, which must be received by Friday 23 March 2012 can be submitted by individuals or teams in four age categories, ranging from 7 – 19. These will be reviewed by a panel of judges and the shortlisted winners will be announced in May 2012.

WHERE DID MIND EXPANDING TECHNOLOGY COME FROM?

Technology develops at a bewildering speed. Today's developers stand on the shoulders of giants. Roger Davies praises a free book that looks at the origins of the ideas that underpin the computing revolution.

"You can't understand where mind-amplifying technology is going unless you understand where it came from." So says Howard Rheingold in his introduction to 'Tools For Thought'. Written in the early 1980s, Rheingold was surveying the onset of the personal computer. Many books tell that story as the work of Jobs, Wozniak, the homebrew clubs and teenagers in garages. Good as those stories are, Rheingold points out, "If it wasn't for people like JCR Licklider, Doug Engelbart, Bob Taylor, Alan Kay, it wouldn't have happened. But their work was rooted in older, equally eccentric, equally visionary, work, so I went back to piece together how Boole and Babbage and Turing and von Neumann created the foundations that the later toolbuilders stood upon to create the future we live in today."

This is a wonderful book, freely available, though tucked away on his website (link in the supplement). It explains the origins of the ideas on which today's complex technologies rest, and it does so in an engaging accessible fashion. Rheingold weaves a story that takes us from the ideas of Babbage and Lovelace's programming, Boolean logic and Turing-type computation to the scientific and technological breakthroughs driven largely by the politics of the post war world. Rheingold has a way of explaining complex concepts in layman's terms without robbing them of their depth. Thus he explains the importance of Information Theory, the early explorations in Artificial Intelligence, network technologies, developments in storage, processing and user interfaces. Most importantly, there is a chapter recognising the work of Seymour Papert and Alan Kay. Their vision of the computer as a child's tool for thought, or, in Rheingoldian terminology, a mind amplifier is a 'must read' for every computing teacher. It ends speculating about developments in immersive worlds and online communities many years before internet use became a widespread.

Chapter three is a succinct explanation of Alan Turing's major contributions to this amazing history. His vision of a Universal Machine, which could be programmed to perform as a 'virtual' machine is remarkable for its clarity as is the unification of his ideas on programming, translation, debugging and machine dialogue. As the book makes clear, Alan Turing stands at the centre of the development of deep, rich, intellectual discipline - computing.

A CALL FOR TURING 100 PARTICIPANTS

Turing100 invite students, teachers and others to be a part of a historic and exciting event. Using the famous Turing test the preliminary phase will start soon and continue till May. Participants will be asked to judge 'entities' on special Turing100 web pages. They will be asked to judge the conversation ability (0=bad to 100=humanlike) of each of the six entities. See the supplement for details.

APPRECIATING TURING'S WORK: TAKING PUPILS TO SEE STATION X

If you have never taken pupils on an educational trip, the Turing centennial provides a obvious opportunity, and Bletchley Park an obvious location. This was the location of the wartime code breaking centre, where Turing played a central role.

The most famous of the codes and ciphers broken at Bletchley Park (known as Station X) were the Enigma machine generated ciphers, but lots of other coding systems used by Hitler and his allies were also broken. Behind the magnificent mansion in the Buckinghamshire countryside, large wooden huts were erected on the lawns of the estate. These became home to the famous codebreakers of the Second World War and one of the birthplaces of modern computing and communications.

A team including Turing made the first break in Enigma in early 1940. More successes followed so that by April they had cracked both the German Army and Luftwaffe ciphers. The process of breaking Enigma was due to a complex electro-mechanical device, designed by Turing, known as The Bombe. The Enigma ciphers were changed daily. The Bombes were operated by an army of Wrens in Hut 11, greatly reducing the odds (and time) to determine the ever changing settings used. As the code breaking work increased, the numbers working at Station X swelled to over 9000 personnel. Additional Bombe units were sited around Buckinghamshire, and by 1944 there were at least 200 in use.

You can take a group around the grounds of Bletchley Park, identifying the surviving huts and learning more of this fascinating story. Other buildings too, are steeped in history. 'Cottage Number 3' was where the first break was achieved, whilst 'The Bungalow', an old fruit store became a think tank for early computer research by mathematicians including Turing, Gordon Welchman, Max Newman and Tommy Flowers. In 1943 the Nazis had developed Lorenz, a semi automatic machine even more complex than Enigma. Newman was convinced the answer to cracking Lorenz ciphers lay in developing a computing machine such as that described by Turing in his pre-war thesis. At the roundabout, some crumbling steps are all that remains of Block F. It was here, in late 1943 that Tommy Flowers housed Colossus, Bletchley Park's greatest success. So successful was Colossus that a Mark II model was designed and six were housed in Block H in 1944. Block H is now the home to The National Museum of Computing (see right). The main exhibition centre is found in Block B whilst other buildings house a range of associated exhibitions.

Pre-booked school visits are welcomed, tailored to different age groups, with Computer Science themed visits organised through the NMoC. Further details can be found on their informative website. See supplement for links.

FIND OUT MORE ABOUT BLETCHLEY PARK FROM CS4FN

Our friends at CS4FN have lots of extra information about Bletchley Park. Whether you want to know how you can help contribute to the restoration, the debate about just which machine can claim to be the first computer, how the Bombe worked, or a host of activities on frequency analysis and code breaking their website has it all. Between them, Bletchley Park, the NMoC and CS4FN websites will provide students with an excellent resource for pre or post visit work.

THE NATIONAL MUSEUM OF COMPUTING

The National Museum of Computing houses the Colossus computer rebuilt by Tony Sale and continues the history of the development of computing from the 1940s to the present day.

Operated separately by the Codes and Ciphers Heritage Trust, a visit is typically made up of a short introduction, tour or workshop and optional presentations. The NMoC is currently developing presentations on

- Computing history
- The development of storage
- The race to get faster
- What exactly is a computer
- The Turing machine

They also have a classroom with a rebuilt BBC computer cluster. They currently offer workshops for post 16 Computing students on

- First Generation Machine Code Programming on PDP8s
- Second Generation Assembly Code Programming on PDP8s
- Third Generation BASIC Programming on BBC micros.

A group may combine these activities with a look at the exhibition in Bletchley Park showing the Enigma machines and the work of the code breakers. Prices are dependent on the age of pupils, but very reasonable. Further details on their website.

CREATING A GRAPH OF A CODE SEQUENCE

As part of A-level Computing we show the students a number of interesting techniques that they could use in their projects, e.g. unlimited precision arithmetic, graphs (of the Sales v Temperature variety), validation, and sub-classing Windows controls.

One useful technique is that of scanning: taking a stream of characters, and breaking it down into its tokens or components. When a GetNextToken function is called a token class (identifier, keyword, symbol, string, etc) is returned, possibly with some additional information, e.g. the name of the identifier, an enumerated value for the keyword, or the value of a number. So, given some code like this:

```
If n <= 5 Then
    n += 25
End If
```

repeated calls to GetNextToken give:

keyword, If	
identifier, n	
symbol, <=	
integer, 5	
keyword, Then	endOfLine
identifier, n	
symbol, +=	
integer, 25	endOfLine
keyword, End	
keyword, If	endOfLine

Scanners are really useful for compilers, interpreters, and free-form data entry. Rather than sprinkle lots of string handling code throughout our program we can just keep on calling the GetNextToken subroutine and decide what to do based on its return value. The 'deciding what to do' process is called parsing and can be introduced to students when looking at defining grammars using Backus Naur Form - yet another link between topics at A-level!

COMPILERS, GRAPHS AND ALAN TURING'S HALTING PROBLEM

Teaching A-level Computing offers plenty of opportunities to make links between different areas of the syllabus. John Stout shows how lexical analysis and graph structures can help students visualise the tricky concept of the 'Halting Problem'.

It would be very useful if, before we ran a program, the IDE could warn about potential problems with our program: not the expected type of syntax error, but the 'Did you mean to do this?' type of error. For example, Visual Studio warns us about using a variable before we've initialised it, not using a variable we've defined, and even paths through a function's code which avoid returning a value.

When we program we'd all like to avoid infinite loops: wouldn't it be great if the IDE could say 'Hey, this program will never stop' BEFORE we run it? This is the core of the Halting Problem: Can we write a program that, when given ANY program and some data, tells us if it will halt or not when run with that data?

So where do graphs come in? Well, think about programs as a directed graph by
1. numbering each line of the program from 1 to n
2. drawing a graph where each line in the program is represented by a vertex
3. join two vertices i, j with an edge if line j of the program is (or could be) the line of code executed immediately after line i (i is the source, and j is the destination).

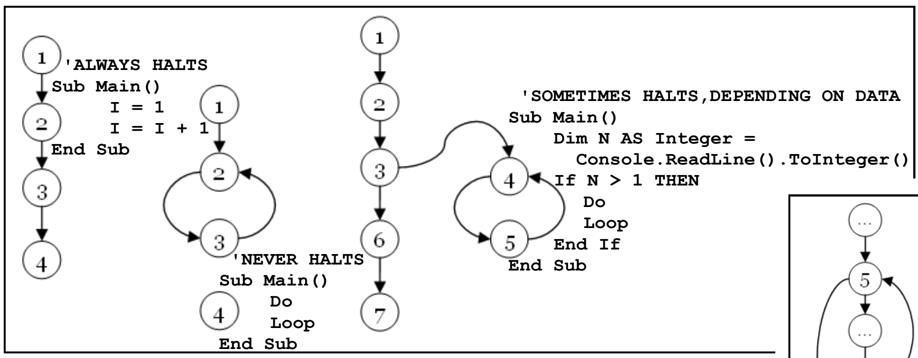
The box above right shows the graphs that represent three sample programs. Notice that the graph feature that distinguishes a program that never halts (or possibly never halts) from one that always halts is the presence of a cycle (representing a programming loop); lines 2 to 3 / 3 to 2 in the second program and 4 to 5,/ 5 to 4 in the third. If there is a way of avoiding (or getting out of a cycle) then a program could halt – depending on the input.

Generating the graph of a program uses a technique called scanning to identify keywords such as If, Do, Loop, End and so on. The sidebar left explains the concept in more detail. The algorithm to produce a graph can be written in pseudo-code as

```
initialise
do until no more lines
    get next program line
    if this line is the destination of a previous line(s) then
        generate one or more edges from their source to this line
    if this line is a source (backwards/forwards in the program) then
        save this line's number
```

Once we have a description of the graph we can draw it using the DOT language. This is a very useful technique for some student projects. See the far sidebar for more details of what is involved. The web supplement contains a simplified version of the scanning / graphing program written in VB. It only handles enough control structures to make its point, and has very little in the way of graceful recovery from errors. Nonetheless, it is a useful program to demonstrate and worth studying.

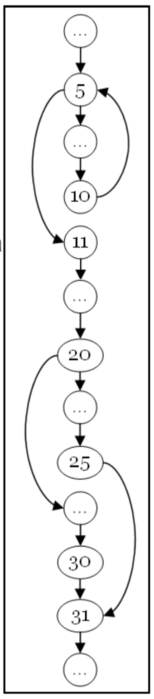
The heart of the program is in the function called graphURL which takes the text of a program (parameter programText) and returns a Google Charts URL to draw the graph of that program. It creates an instance of the Scanner class on the program text, then scans through the text using, in essence, the algorithm



```
'ALWAYS HALTS
Sub Main()
        I = 1
        I = I + 1
End Sub
```

```
'NEVER HALTS
Sub Main()
       Do
       Loop
End Sub
```

```
'SOMETIMES HALTS,DEPENDING ON DATA
Sub Main()
     Dim N AS Integer =
        Console.ReadLine().ToInteger()
     If N > 1 THEN
        Do
        Loop
     End If
End Sub
```



outlined previously. As always the devil is in the details, but the diagram right demonstrates the principle. For example, when finding a For loop (at line 5 say), it stacks a CodeNode object of type "For" and line number 5, then generates an edge from line 5 to line 6, and continues through the code inside the For / Next loop until it gets to a Next (say at line 10), when it generates an edge back to line 5 and an edge from line 5 to line 11. Similarly, when finding an If / Else / End If structure (at line 20) it stacks a CodeNode object of type "If" at line number 20, generates an edge from line 20 to line 21, and continues through the code of the True branch. At the Else (line 25) an edge is generated from line 20 to line 26, and then finally at the End If (line 30) edges are generated from line 25 to 31 and from 30 to 31. Stacks have to be used since programs can nest control structures of different types within each other.

Let's get back to the halting problem. Students can find this difficult to come to terms with. Turing proved that writing a program to determine whether any program/data combination does or doesn't halt is an impossible task. The argument goes like this:

1. Imagine that we can write such a program, call it HaltChecker. If we give it ANY program (P), and the data that P is to be tested with (D), it ALWAYS returns True or False (True if program P does halt when run with data D, False when program P doesn't halt when run with data D). This can be represented as the graph shown on the right. Notice that HaltChecker ALWAYS returns a value, i.e., it NEVER gets stuck in a cycle.

2. Now we write another program, let's call it HaltCheckerPlus, containing the HaltChecker program as a subroutine, and using it in a sneaky way: HaltCheckerPlus doesn't halt if HaltChecker says the program it's testing does halt, and it halts if HaltChecker says the program it's testing doesn't halt. Again see the graph on the right to visualise this.

3. Finally, let's ask ourselves what happens if we run HaltCheckerPlus, with HaltCheckerPlus as the program it's testing and as the data?

Answer: if HaltCheckerPlus would halt, HaltCheckerPlus doesn't halt, and if HaltCheckerPlus wouldn't halt, HaltCheckerPlus does halt!

4. We clearly have a paradox, yet each step of our argument is logical. The only thing that's wrong is our first assumption: that we could write such a HaltChecker program in the first place. This is known as a proof by reductio ad absurdum.

The proof as outlined not only proves that we can't write a HaltChecker program that works for EVERY program, but it also shows that there are programs (well, at least one) we can define but that we cannot write. The Halting Problem is a difficult idea to grasp but by showing the programs as graphs helps students to visualise the proof, whilst also drawing links between topics. *John Stout*

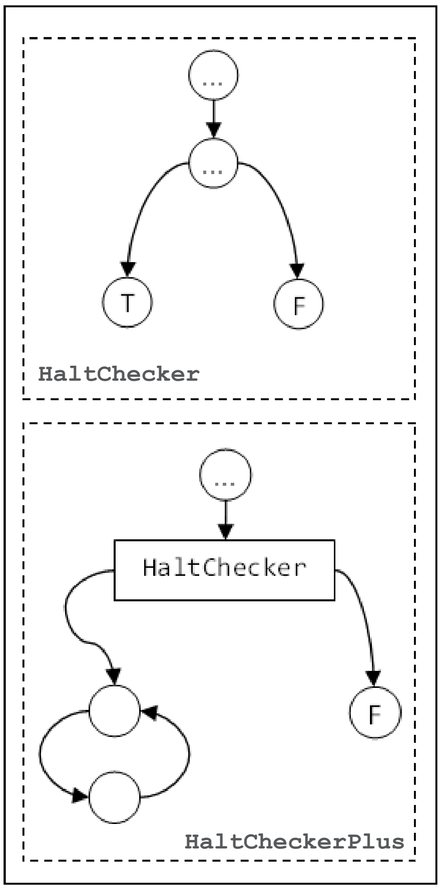
USING DOT LANGUAGE TO CONSTRUCT GRAPHS

Graphviz is free, open source graph visualisation software. You'll find the link to their website in the supplement. Graphviz programs take a graph description in simple text language called DOT. To draw the 'Sometimes Halts' graph, above left, using DOT we need to generate a text file containing:

```
digraph SometimesHalts {
    1 -> 2 -> 3 -> 6 ->
7; 3 -> 4 -> 5 -> 4;
}
```

The description is passed to the DOT language processor. An alternative is to generate a URL for Google Charts like this:
`chart.googleapis.com/chart?cht=gv&chl=digraph{1->2->3->6->7;3->4->5->4}&chs=150x150`

DOT has lots of other options for controlling layout, fonts etc but for our purposes this is all we need.



DR SEUSS, POETRY AND PROOF: SCOOPING THE LOOP SNOOPER

If you, or your students are still struggling to comprehend the undecideability of the Halting Problem, try this wonderful proof written in the style of 'Dr Seuss' by Geoffrey K Pullum of Edinburgh University.

No general procedure for bug checks succeeds.
Now, I won't just assert that, I'll show where it leads:
I will prove that although you might work till you drop,
you cannot tell if computation will stop.

For imagine we have a procedure called P
that for specified input permits you to see
whether specified source code, with all of its faults,
defines a routine that eventually halts.

You feed in your program, with suitable data,
and P gets to work, and a little while later
(in finite compute time) correctly infers
whether infinite looping behaviour occurs.

If there will be no looping, then P prints out 'Good.'
That means work on this input will halt, as it should.
But if it detects an unstoppable loop,
then P reports 'Bad!' --- which means you're in the soup.

Well, the truth is that P cannot possibly be,
because if you wrote it and gave it to me,
I could use it to set up a logical bind
that would shatter your reason and scramble your mind.

Here's the trick that I'll use -- and it's simple to do.
I'll define a procedure, which I will call Q,
that will use P's predictions of halting success
to stir up a terrible logical mess.

For a specified program, say A, one supplies,
the first step of this program called Q I devise
is to find out from P what's the right thing to say
of the looping behaviour of A run on A.

If P's answer is 'Bad!', Q will suddenly stop.
But otherwise, Q will go back to the top,
and start off again, looping endlessly back,
till the universe dies and turns frozen and black.

And this program called Q wouldn't stay on the shelf;
I would ask it to forecast its run on itself.
When it reads its own source code, just what will it do?
What's the looping behaviour of Q run on Q?

If P warns of infinite loops, Q will quit;
yet P is supposed to speak truly of it!
And if Q's going to quit, then P should say 'Good.'
Which makes Q start to loop! (P denied that it would.)

No matter how P might perform, Q will scoop it:
Q uses P's output to make P look stupid.
Whatever P says, it cannot predict Q:
P is right when it's wrong, and is false when it's true!

I've created a paradox, neat as can be ---
and simply by using your putative P.
When you posited P you stepped into a snare;
Your assumption has led you right into my lair.

So where can this argument possibly go?
I don't have to tell you; I'm sure you must know.
A reductio: There cannot possibly be
a procedure that acts like the mythical P.

You can never find general mechanical means
for predicting the acts of computing machines;
it's something that cannot be done. So we users
must find our own bugs. Our computers are losers!

THE TEACHING OF ARTIFICIAL INTELLIGENCE AS SCIENCE

Aaron Sloman encourages teachers to get involved in resurrecting the teaching of AI in schools. Inspired by Turing's ideas on morphogenesis, he argues for extending the idea to a concept of meta-morphogenesis.

Some young learners decide not to take courses in programming because they feel other scientific subjects have more academic value. One way of countering this is to emphasise the deep scientific content in the science of computation. As an alternative, which may be of interest to a different group of learners, a subgroup of members of CAS has begun to explore ways of introducing AI programming as a way of doing science, e.g. modelling human reasoning, learning, planning, or use of language, by using languages specially designed to support such experiments in "thinky" programming. Many AI researchers are primarily interested in AI as the science of intelligent systems.

One of Alan Turing's highly influential papers, The Chemical Basis Of Morphogenesis, put forward conjectures about how interactions among molecules in a developing plant or animal could eventually give rise to large scale patterns, such as spots on a leopard, or the spiral patterns of fir cones. We can extend this idea. In many animals, not only new physical structure, but also new information processing capabilities emerge during development, including perception, motor control, use of language, abilities to reason and plan and social competences. Moreover the learning competences also change: what can be leant by human infants, 5, 10, 15 year olds and so on changes dramatically.

We can express this by saying that in addition to morphogenesis of information processing capabilities there is also morphogenesis of morphogenesis (learning new ways to learn, for example). We can label this "meta-morphogenesis" (MM). In addition, what different species can learn, and what they can learn to learn, changes across evolutionary time scales, another example of meta-morphogenesis (actually meta-meta-morphogenesis - MMM). As far as I know, there has been very little explicit study of how to model varieties of MM, although some beginnings are to be found in AI research on learning and evolutionary computation. This appears to be a large new field for expansion of the science of computation. In future we may have to learn to build useful systems that undergo meta-morphogenesis partly under the influence of the environment in which they are applied. Perhaps we should start educating young learners to think about this. The AI group are collecting examples and links on this topic. Further links in the supplement. *Aaron Sloman*

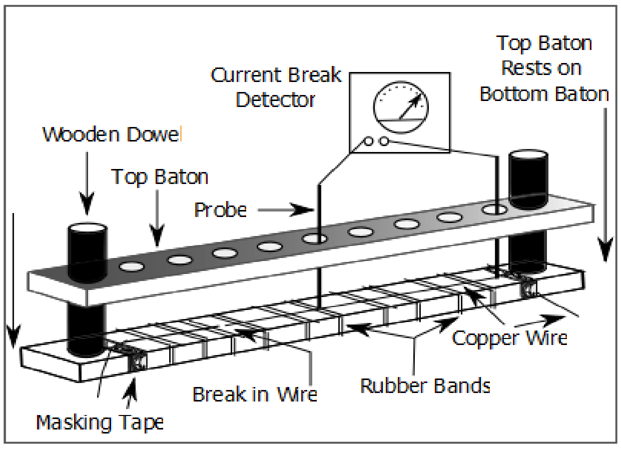
Aaron Sloman has produced a podcast introducing rule-based programs based on an AI language, showing some of the power of pattern-matching for structure-manipulation — crucial for systems that think, see, learn, solve problems, make plans, communicate, etc. The video includes interactions with a running program whose capabilities are extended during use, as it acquires more information from a human. The program can be run remotely via a virtual machine provided by Lee Gillam. To run the code (and other tools) locally on a Linux system contact Aaron.

WHEN SIZE DOES MATTER (PART TWO): GETTING TO GRIPS WITH THE NOTION OF ALGORITHMIC COMPLEXITY

The AQA Computing specification for A2 requires candidates to have some appreciation of the complexity of a problem defined as the growth rate of the algorithm which solves the problem, i.e. its big O complexity. Following on from the article in the last issue, Chair of Examiners, Kevin Bond suggests some practical ways to explain O(Log$_2$n) complexity to your classes.

You don't need a computer to demonstrate log2 n complexity. Consider a low voltage light at the bottom of a garden. The cabling is in the ground. The light isn't getting current, even though the power supply in the shed is fine. The fault must be located without digging up too much of the garden. How? A technique called the split-half method can quickly locate the fault. A hole (and test) is made halfway between the shed and the light. If there is current between the shed and the hole, the break must be between the hole and the light. Applying the technique again, you make a second test, half-way between the hole and the light and so on. In this way the number of holes is minimized and so also the time taken.

This can be simulated in the classroom. Two batons slotted together on wooden dowels cover a length of bare copper wire strapped to the lower baton with masking tape and several elastic bands. The copper wire is cut and separated slightly as shown in the diagram so that the small gap is invisible through the holes in the top baton. Nine holes are spaced evenly with the two end holes approximately 64 cm apart (at 0, 8, 16, 24, 32, 40, 48, 56 and 64cm). A simpler version can be constructed with just one baton. A strip of masking tape with holes made at fixed intervals covers the length of copper wire. Sections of the wire are tested as shown – the full length, then one half then one quarter and so on. Any circuit continuity tester can be used. If you borrow a multi-meter from the science department ensure it can measure resistance (unit Ω).



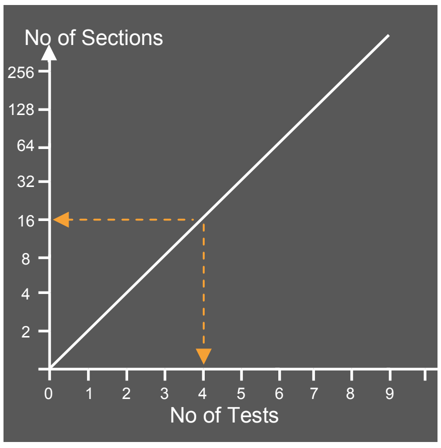
Let's say that the break is between 8cm and 16cm. With the probes of the current break detector in holes 0cm and 64cm the instrument should indicate a high resistance to current flow (indicating a break). Now move the 64cm probe to the 32cm hole. The meter should still indicate a high resistance to current flow. Next place the probes in 0cm and 16cm where a high reading should once again be obtained. Finally, test the last section 0cm and 8cm. This time the reading should be low, indicating an unbroken circuit between 0 and 8cm. Although there are 8 sections in total, the split-half technique has reduced the number of tests (needed to locate the break) to 3. The same answer would be obtained if the break was in any of the other sections, e.g. 16 - 32. Try this for yourself. Pick a section at random for the break and count the number of tests required.

How many tests would be needed to find a break between 0 and 4cm, with the wire split into 16 sections? The answer is 4 because sections will need to be tested 0-32, 0-16, 0-8 and finally 0-4. Five tests would need to be performed to find a single break among 32 sections and six tests for 64 sections. The number of tests increases by one when the number of sections is doubled. This is shown in the table above. The number of tests is the same as the exponent when the number of sections is expressed as a power of 2, e.g. 16 expressed as a power of 2 is 2^4. The exponent in this example is 4. In conclusion:

Number of Sections	Number of Sections$_2$	Number of Tests
128	2^7	7
64	2^6	6
32	2^5	5
16	2^4	4
8	2^3	3
4	2^2	2
2	2^1	1

No of Tests = Log$_2$ No of Sections

Therefore, the growth rate of number of tests for this technique is O(log$_2$ n) where n is the number of sections. This is shown in the line graph below where the number of tests grows by one each time the number of sections is doubled. An explanation of the maths behind this can be found in the supplement.



IBM PARTNERS WITH VITAL TO PROVIDE IT INSIGHTS

IBM's Smarter Planet initiative identifies the need for a smarter use of today's technology. Through the use of RFID tags and sensors, data can be harvested and manipulated to provide the necessary output to allow us to make smarter decisions which will have a positive effect both in terms of profitability and environmentally. IBM's Colleen Haffey (Software IT Architect, IBM), Sophie Bialaszewski (Community Programmes, IBM) and Sue Nieland (Course Developer Vital Specialist, E-Skills) facilitated an initial teachers' workshop at Park House School & Sports College, Newbury.

Presentations from Tony Horrocks (Smarter Planet), Alan Flack (Wimbledon technologies) and Chris Bray (Careers in IT) provided fantastic contexts for teaching students about smart systems design and development including the use of augmented technology, identifying usability of digital natives and ensuring that systems are instrumented, interconnected and intelligent. Following the pilots a series of further workshops have been provided at different venues. Lesson resources have been produced to help enrich computing at KS3. The series of connected lessons aim to get them to think differently about systems design and development. They take students through the process of identifying problems they experience inside and outside of school, identifying new technologies, designing apps for intelligent use and presenting their solutions. Download details are in the web supplement. *Pete Marshman*

SUSSEX HUB EXPLORES APP DEVELOPMENT WITH YOUSRC

At the Autumn meeting of Sussex CAS, Paul Clarke from Previca introduced an online development environment he first developed in order to enable his daughter to explore and write code.

YOUSRC (pronounced "You source") uses a very simple programming language called ELC (named after the young YOUSRC coder Emma Louise Clarke) that takes its roots from many of the common programming languages around. Because of this it is a very good starting point. A language from which people can move on to more and more complex and powerful environments. It is simple and the students can control their own learning and go at their own pace. By having an Android player they are able to go home and play their apps on their Android phones....no Blackberry or Apple as of yet.

We have used it in our Year 8 classes and it has been a great success. Even though we didn't manage to get anything completed for the competition there is always next year! Paul is always available for any request, even silly ones. It is a great way to the get the students using code and the whole development cycle. Following YouSrc we moved on to Alice and student understanding has increased dramatically in comparison to the groups that hadn't done YouSrc before. They are different but it opens up access to many students who believe that they "can't do computers". Thanks Paul. *Genevieve Smith-Nunes*

GOOGLE CS4HS GRANTS 2012

Google's grant program, Computer Science for High Schools (CS4HS) provides funding to universities who work with schools to engage students in computer science. In 2011, three UK universities were awarded funding. Queen Mary University received continued support for their CS4FN program, the University of Kent for Greenfoot, and the University of Manchester for the UK School Computer Animation Competition. Applications for 2012 close mid February. Further details in the supplement.

CAS hubs continue to grow, as teachers appreciate informal opportunities, like the Sussex meet above. Six new hubs held their first meetings during November and December 2011: South Coast (Poole), Three Counties (Malvern), West Yorkshire (Leeds), NE Scotland (Elgin), West Midlands (Birmingham) and Hampshire (Alresford). Other CAS teacher hubs that met last half term were Surrey, Norfolk, Bristol, Thames Valley and Bucks. They are proving an invaluable source of support. There are still some areas not represented though. Please get in touch with me (see supplement for details) if you teach/live in the following areas and would like our help in setting up a hub: Cumbria, Northumberland, Shropshire, Staffordshire, Lincolnshire, East Yorkshire, South Yorks/Notts, Devon/Cornwall and Kent. ***Claire Davenport***

WOMEN IN COMPUTING: SCHOOL PUPILS ENJOY INSPIRATIONAL DAY

When a group of Year 10 students from Bay House School, Alverstoke in Hampshire attended a 'Women In Computing' event at the University of Oxford, Computer Science Department they had no idea what to expect.

None of us knew what to expect from the day, but we were excited about the experience. The first activity was a lecture about what computer science is. We were given an insight into how a computer multiplies numbers, which was really interesting because what seemed like a simple process to us was actually quite complicated for the computer. Computers don't have the same type of thought process that we have, so they have to be told every individual thing to do which makes doing the simplest of tasks complex. We also watched a presentation on 'Flying Helicopters – Research into Unmanned Aerial Vehicles (UAVs)'. Computer science doesn't always mean sitting in an office - it can save lives too. The helicopters were programmed to find people after natural disasters.

We heard about a new course in Computer Science and Philosophy. I had never thought about these two together, but as the professor explained, it made more sense in my head. Philosophers answer questions about how we think and computer scientists are trying to replicate the way we think in machines. Trying to understand why we think the way we do could help us replicate thought processes and teach computers how to think. This is just one example of how the two subjects cross over.

After lunch we discovered more links with other subjects. For example a woman talked to us about how computer science and biology is used to make models of the human body and how these models are being used to test medicines without harming people.

In the computer lab we used a program called GeomLab, which allowed us to create pictures on the screen. The configuration of the pictures allowed us to resize and rotate the images. We were taught how to assign functions which showed us how simple and fun programming can be and it inspired us to download the program later to experiment more.

Finally we were taken around St John 's College. We saw students studying in groups and got an insight into the work ethic, the friends, and fun that come with studying at university. We all learned a lot. It inspired us to look more into computer science and consider it as one of our options for later on in our lives. *April Selby*

GETTING TO GRIPS WITH GCSE COMPUTING

A course for ICT teachers wishing to extend their skills to teach GCSE Computing will be offered at Anglia Ruskin University, Chelmsford. The evening course will run for 10 weeks, from January 17 to March 27. The theory to be covered will include:

- binary and hexadecimal numbers
- computer architecture
- representing sound and graphics
- networking
- structure of the internet
- algorithms and problem-solving

The programming language used will be Python, using resources already developed at last year's summer school, as reported in the last issue. It will cover:

- variables and assignment
- selection and iteration
- arrays and other data structures
- file-handling
- working with an SQLite database

The course has been offered in response to the requests from ICT teachers to become familiar with the background required to deliver the new OCR GCSE Computing, which is proving very popular. Like the Python Summer School, it is hoped to make the resources available online for teachers in other areas to access.

Sue Sentance

GOOGLE TRAILBLAZER PRIZES AT BIG BANG FAIR 2012

As part of Google's ongoing initiatives to promote and encourage the study of computer science, we will be attending the Big Bang Fair in Birmingham in March 2012. We hope to excite students with some hands on experience and engage them through challenging analytical problems. Google will also award two prizes for the best computer science projects of 2012. As well as receiving £500 and a certificate, each of the two computing trailblazers (plus a guardian if necessary) will have the opportunity to spend two days at one of the Google research sites in Europe where they will be able to take part in tours, mentoring from Google Engineers, workshops and get involved in some hands-on work! Further links to Trailblazer details in the supplement. ***Niall Byrne***

THINK COMPUTER SCIENCE

Think Computer Science is an annual event Microsoft Research hold to showcase the work of computer science researchers and to enthuse Year 8/9 students about the field of computer science. The 8th annual event was held at the Imperial War Museum, Duxford on December 7th just as this issue was finalised. More details in the next issue.

A YEAR OF RAPID PROGRESS FOR CAS MEMBERS IN WALES

CAS has made significant inroads in raising the profile of Computing in Wales since the inception of the first CAS Hub in September 2010. We have focused on widening and developing our network of Computing/ICT teachers, and have been active at national policy level by working with the Welsh Government and different examination boards.

Computing is certainly on the Welsh Government's radar from both education and economic renewal perspectives. It also underpins their wider "Delivering a Digital Wales" framework. Alongside the CAS contribution to national-level policy consultations, we have been working hard to raise the perception of Computing as an academic discipline distinct from digital literacy. We have also focused on Computing as a core STEM subject, as well as how it underpins science and research strategy. We have met with a number of Welsh MPs and AMs, as well as policy advisors within the Department for Education and Skills, and will continue to spread the message! We have also had good coverage from BBC Wales.

Part of this success is down to CAS Wales' strategic collaboration with **Technocamps**, a £6 million project led by Swansea University that aims to inspire young people aged 11-19 through a range of exciting computing-based workshops on topics such as robotics, game development, animation and digital forensics. Funded by the European Social Fund through the Welsh Government, it has the long term goal of encouraging young people to pursue careers in Computing and related STEM areas that will drive economic growth in Wales. Together with Technocamps, we jointly hosted the successful inaugural CAS Wales Conference at Swansea University in July 2011.

There is much to be done; but by working closely with Technocamps we feel we can advance the Computing agenda in Wales, and be able to report on exciting developments in these pages on a regular basis. *Tom Crick*

KIDS' COMPUTER WORKSHOPS A HIT WITH TECHNOCAMPS

In mid October, 120 students from Swansea became the first pupils to experience a Technocamps Workshop. By the start of 2012, a thousand more school children will have attended Technocamps Workshops throughout Wales.

First through the door were pupils from Morriston and Cefn Hengoed Schools. The interactive Workshops, delivered on campus at Swansea University, helped the pupils get to grips with computer programming in a practical and creative way. The pupils were from different year groups but enjoyed the same experience. Each Workshop began with hands-on activities that encourage the pupils to think about the importance of clear, precise instructions, which is fundamental to programming. The first activity involves the participants dividing their tables into two teams and secretly designing a mascot; they then have to give the opposite team instructions on how to recreate it without showing them the initial design. The second activity involves one pupil coming to the front of the room and secretly drawing a picture; a second pupil looks at the image and gives instructions to the rest of the class on how to reproduce it.

The pupils were then introduced to programming: Scratch for the 11- to 13-year-olds and Alice for the 13- to 16-year-olds. Each pupil has access to a MacBook Pro with which they are guided to create their own individual animations and games. The Technocamps experience doesn't stop there; follow-on material is provided on a weekly basis for use in the fun and stimulating atmosphere of an extracurricular Technoclub to continue the pupils' development. Each group that attends a Workshop will return some months later for a further day with an emphasis on exploring the activities carried out in their Technoclubs.

We are currently developing further Workshops on diverse topics such as cryptography, robotics, computer forensics, algorithmics, and general problem solving through computational thinking. Topics are typically motivated by the research expertise at the partner Institutions (Aberystwyth, Bangor, Glamorgan and Swansea Universities). Furthermore, each Workshop includes an inspirational talk. In October these were given by Dr Philip Legg of Swansea University's Sports Science Department who discussed app development and marketing, and demonstrated his MatchPad App which was at that time being used by the Welsh Rugby team at the World Cup. Through these inspirational talks, pupils are given an insight into exciting projects taking place within universities and industry across Wales, and are hopefully motivated to consider career opportunities in computing. *Faron Moller*

SEVERAL ORGANISATIONS JOIN FORCES AS CAS SCOTLAND IS BORN

The Scottish Institute of Computing Educationalists (SIoCE) has joined forces with other bodies to form CAS Scotland. Kate Farrell, chair of the new organisation, reports on the progress made during a very busy first term.

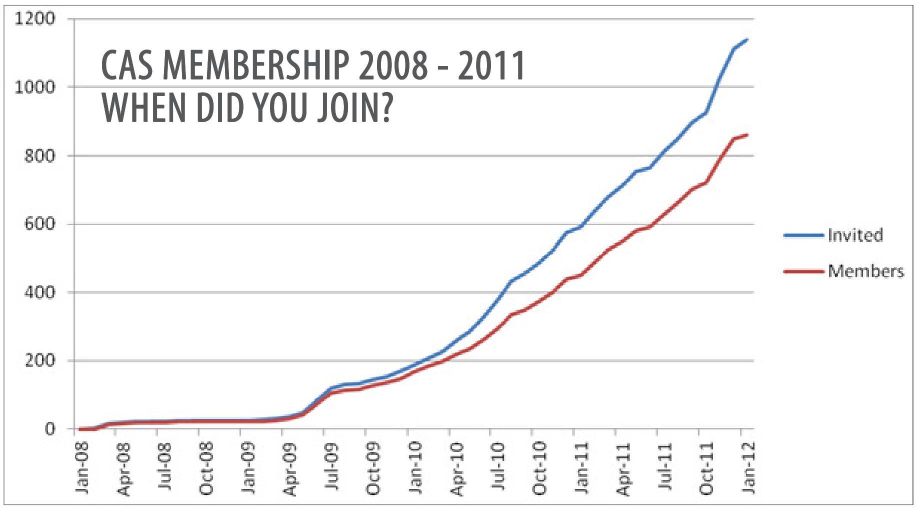
CAS Scotland has have been working with and talking to the Royal Society of Edinburgh, BCS, The Chartered Institute for IT, Computing at School, Scottish Informatics and Computer Science Alliance (Scottish Universities) and various industry partners, as well as participating in debates on the future of Glow and ICT in Scottish Education. Oh, and doing some teaching when we have a chance!

The Royal Society of Edinburgh & BCS, The Chartered Institute for IT have seconded Jeremy Scott (Principal Teacher of Computing at George Heriot's School in Edinburgh) to exemplify some of the Computing and Information Science technologies outcomes in Curriculum for Excellence. The focus for the project will be mainly Level 3 and Level 4 experiences and outcomes. As part of this work there is a new group on CompEdNet to gather the opinions and suggestions of members for the project. Links to register can be found in the supplement. This project will generate a range of materials and resources for CfE Computing courses appropriate for 1st to 3rd year pupils. There will be three packs developed and released in 2012.

The Scottish Qualifications Authority is currently developing new qualifications to replace current ones at all levels as part of Curriculum for Excellence. The SQA are publishing draft documentation as it becomes available and have just finished consulting on the unit specifications for the new National 4 and 5 qualifications. These will be in our schools in 2013.

As part of the Turing Centenary the Royal Society and University of Edinburgh are running the T100 TwitTest Schools Project. Pupils have to decide which tweets are real and which have been created by an AI software bot. This Turing Test for the next generation will allow pupils, parents and teachers to explore just what intelligence is and how we really know if someone, or something, is intelligent. Again, links can be found in the supplement.

Finally, a brief mention of the first CAS Scotland hub meeting, hosted by Claire Griffiths, on 15 Nov Moray, NE Scotland. Eleven people attended to hear guest speaker and local website designer Sam Hampton-Smith - the first of many, we hope. *Kate Farrell*



EXCITING YEAR AHEAD AS CAS GROUP GROWS

The numbers of people signing up for CAS grows weekly (see chart below). Around 70% of the members are active school teachers; but there is a solid core of IT professionals, software developers, parents, governors and university academics. There is significant interest in encouraging computing, as a subject, in our schools. The CAS message is getting heard! CAS is a grassroots organisation - it is you, the teachers, who will make the difference.

This year will see significant changes to the school curriculum. The outcome of the National Curriculum Review is not yet known but there are some indications. There are likely to be changes to the accreditation of vocational courses and their contribution to league table indicators. ICT, the subject largely responsible for the huge uptake in vocational qualifications has been criticised in many quarters. The indication is that the government will favour a slimmed down curriculum, with flexibility for schools to decide what else to teach beyond.

There is a growing awareness that Computing has been neglected in many schools (see the recommendations of the Livingstone Hope Report and the government's response). ICT teachers could feel threatened by impending changes but an increasing number are recognising the opportunities available to develop computing. These exciting opportunities increase the intellectual standing of the department, inspire children and lay the foundation for future progression into KS4 computing courses. CAS is planning ways to support such teachers through the development of the on-line community, resource repository and training courses. Exciting times lie ahead! *Simon Humphreys*

A PAUSE FOR THOUGHT

CAPTCHA - Completely Automated PublicTuring test to tell Computers and Humans Apart - is a challenge-response test. They are used to prevent automated software from posting to blogs, forums and wikis. Computers are presumed unable to solve CAPTCHAs so users entering a correct answer are presumed human. It's known as a reverse Turing test, since a Turing test is administered by a human targeting a machine while CAPTCHA is administered by a machine targeting a human.

A CAPTCHA may ask users to identify a distorted image. Some make them more accessible using obvious questions ("The 2nd letter in "frogs" is?"). Image recognition CAPTCHAs use algorithms involving extraction of colour or other special point features, which cannot be correctly extracted after distortion, whilst still recognizable to humans. Imagination CAPTCHAs create images by applying a sequence of randomized distortions on the original images.

Approaches available for cracking CAPTCHAs include exploiting bugs in the implementation, improving character recognition software, or using humans to process tests. Spammers pay up to $1.20 per 1,000 solved CAPTCHAs to companies in some developing nations. Others attempt to beat visual CAPTCHAs by creating programs that can do Pre-processing - to remove background noise, Segmentation - to split the image into single character regions, and Classification - identifying characters in regions. Computers can relatively easily pre-process and classify, however humans remain better at segmentation. So, if you want to devise a program to beat CAPTCHAs, concentrate on segmentation. *Lyndsay Hope*

FREE MAGAZINES FOR PUPILS FROM CS4FN

Most readers will be aware of the excellent free magazine for computing students but CS4FN have also extended their output to reach pupils drawn to computing through other areas.

The team behind CS4FN has just released a new free magazine: Electronic Engineering For Fun. It casts their eye for a story on electronics and physical computing. The aim of the new magazine is to enthuse students about electronic engineering with articles about real leading-edge research on gadgets, networks, chip design, robots, satellites and technology. Inside the first issue students will find stories about creating their own gadgets, Nikola Tesla, researchers making baby robots, and computer chips embedded in tattoos.

Last term we also distributed Issue 3 of Audio. Audio engineers want to create new sounds and even new instruments to make them. They are responsible for writing systems that, for example, allow out-of-tune pop prima donnas to actually sound great. Now sound has gone electronic it opens up new ways for creating all sorts of things—not just music.

Many readers will receive SwitchedOn with their class set of CS4FN—the original publication supported by a wonderful website. If not, you can order single copies or class sets of any of our magazines for your students at no charge. For more information just visit the CS4FN website. *Jonathan Black.*

COMPUTING AT SCHOOL

EDUCATE · ENGAGE · ENCOURAGE

CAS...JOIN CAS...JOIN CAS...JOIN C

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. Our goal is to put the fun back into computing at school. Will you help us? Simply mail:

membership @ computingatschool.org.uk

Many thanks to the following for help with this issue of SWITCHEDON : *Clive Beale, Jonathan Black, Kevin Bond, Niall Byrne, Tom Crick, Claire Davenport, Roger Davies, Peter Donaldson, Kate Farrell, Graham Hastings, Lyndsay Hope, Simon Humphreys, Peter Marshman, Faron Moller, Simon Peyton-Jones, April Selby, Sue Sentance, Huma Shah, Aaron Sloman, Genevieve Smith-Nunes, John Stout, Andrew Tringham, Liz Upton.* *Front page image © The Bletchley Park Trust*

SEE THE SWITCHEDON WEB SUPPLEMENT FOR MORE LINKS AND EXTRA INFORMATION

www.computingatschool.org.uk

Computing At School are supported and endorsed by:

bcs The Chartered Institute for IT Enabling the information society · Microsoft Research · Google · CPHC The Council of Professors and Heads of Computing · vital