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

# SWITCHEDON

COMPUTING AT SCHOOL NEWSLETTER AUTUMN 2013

# THE PRIMARY CHALLENGE

The new national curriculum makes clear the ambition to "catch 'em young". Computing is back on the curriculum for primary and secondary pupils alike. We know that young children can, given the right context, think in computational terms. Seymour Papert, the father of Logo, pointed to the explosive potential of children, computers and powerful ideas way back in the 1980's. A lot has changed since then, but many of his insights remain true. Indeed, the development of new, engaging visual languages, epitomised by the ready adoption of Scratch by many schools suggests children of the twenty first century will develop further and faster than their counterparts from the earlier Logo generation. Whether that is true will depend on the degree to which the pedagogy that underpinned Papert's 'constructionism' is rediscovered, disseminated and developed. The pedagogical challenge is the primary challenge now facing those who have so successfully fought to establish Computing in schools.

The first half of this issue is focused on encouraging developments in primary schools. It teems with stories of teachers and pupils trying things and reflecting on what works. There is no doubt that many of the new visual programming resources provide hugely motivating environments for pupils. Reading the stories and seeing what is already happening makes you wonder just what is possible!

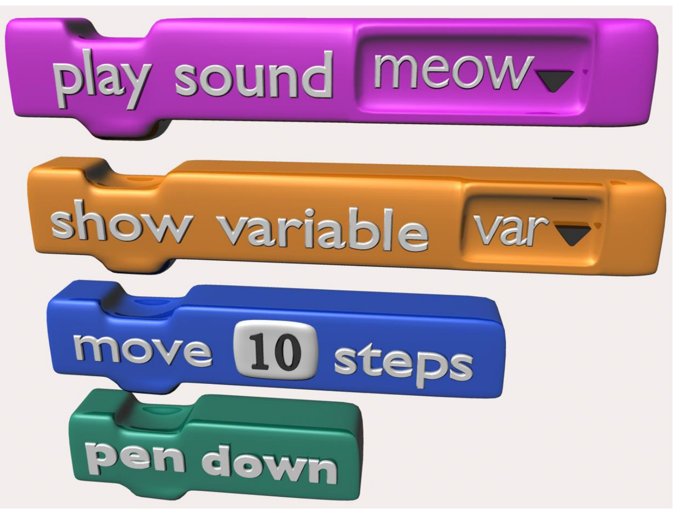


Image by kind permission of Krista Shapton (http://www.kshapton.com)

## INSIDE THIS ISSUE

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

bcs Academy of Computing

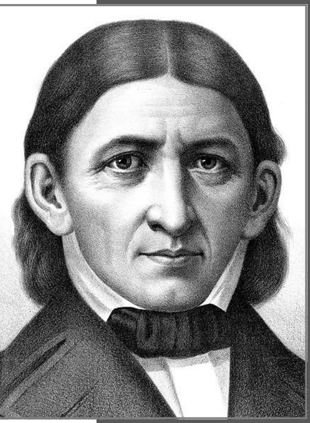
ISSN: 2050 -1277 (online) 2050 -1269 (print)

# COMPUTING: IT'S NOT JUST WHAT WE TEACH, BUT HOW WE TEACH IT

**With a new programme of study for computing there is at least a framework for thinking about what we teach. It's now time to turn our attention to the question of how we teach computing argues Miles Berry, Senior Lecturer at The University of Roehampton.**

## THE ENDURING LEGACY OF FREDRICH FROEBEL

Froebel is best known for his pioneering work in early childhood education, specifically the invention of the Kindergarten, literally, 'children's garden'. It's absolutely no coincidence that the semi-ubiquitous primary programming toolkit, Scratch, owes its origins to Mitch Resnick's Lifelong ***Kindergarten*** Group at MIT. As Resnick puts it: “We are inspired by the ways children learn in kindergarten: when they create pictures with finger paint, they learn how colors mix together; when they create castles with wooden blocks, they learn about structures and stability. We want to extend this kindergarten style of learning, so that learners of all ages continue to learn through a process of designing, creating, experimenting, and exploring.”



The main feature (right) looks at four key elements in Froebels educational philosophy. There are, I'm sure, many other insights to be gained into a pedagogy of primary computing from Froebel's pioneering work, as well as the ideas of other educationalists including Maria Montessori (both of Google's founders attended Montessori schools, incidentally), John Dewey, Jean Piaget, Loris Malaguzzi and Seymour Papert.

A fresh curriculum gives us a chance to think about a fresh pedagogy. However, I don't think there's any need to start from, er, scratch here, as looking back into the history of primary education can give some powerful insights into how we might best move forward.

One source that's particularly worth investigating is the work of the 19th century German educationalist Friedrich Froebel. I'd like to pick out a few aspects of Froebel's vision for the Kindergarten and think about applying those to teaching computing, particularly, but not only, using Scratch.

**The garden:** One of the things which set Froebel's Kindergartens apart from other schools of his day was the emphasis on providing children with an incredibly rich environment in which to learn, not so much through being taught as through purposeful exploration and discovery. We've the chance to do the same in the virtual realm too: providing a diverse collection of devices, software and curated sites might well be sufficient for much meaningful learning to take place, especially if we are on hand to provide the motivation and challenge to boost our pupil's natural curiosity. The benefits of a rich virtual environment are so evident on the Scratch website itself. Whilst many might argue for the superiority of BYOB/Snap!, Logo or even Python, the opportunities for peer to peer learning are so great, so readily available with the Scratch community, that they make it easy for children to pursue their own individual and shared interests, learning from and building on others' algorithms and code.

It should be admitted that the natural world is central to Froebel's philosophy, and this might seem at odds with a subject that seems so focussed on technology. I'm not sure that this distinction is helpful, as I think there's a strong case for CS as the 'zeroth' science, acknowledging that there's something absolutely fundamental about the difference between what is and what is not computable, the notion of information as an emergent property of organised matter, and the idea of CS as a lens through which to understand natural systems, as we had in the earlier draft of the programme of study.

**Building blocks:** Froebel came up with a sequence of gifts: carefully crafted and carefully sequenced collections of objects, from brightly coloured yarn balls through to complex construction sets. Probably best known of the gifts are the sets of geometric building blocks, 19th century pre-cursors to Meccano, Lego and Stickle Bricks. Through playing with these, children discover some of the properties of shape, space and matter: they learn how  structures work, they express themselves creatively, they develop fine motor skills, they work collaboratively. Translating this to the online world, in Scratch (and other block based programming languages) children can learn through play and experiment about how programs work; better still, they get to fit the pieces together to make their own programs, and have the crucial experience of fixing these when they don't work. It grieves me to see classes copying down Scratch programs from a worksheet or the IWB, when there's so much more that can be learnt from structured and scaffolded creative play with these virtual building blocks. Imagine a reception class where all the children have to copy the teacher's building block creation: is that

Johann Sperl's 1885 painting 'Kindergarten'

## THE FREEDOM TO PLAY OR PLAYING GAMES?

There's a difference between play and playing games. There's more freedom in the former, there are rules and objectives in the latter. The latter also translates quite well into school terms, where agreed rules and objectives are not unheard of, hence, I think, memes such as game based learning and gamification. Rules and objectives fit well, too, with programming, and ideas such as interactivity, feedback, challenge, resilience, progression and flow, which its advocated claim for game based learning, seem to apply just as well to programming: coding is the new gaming? Perhaps. But even if not, I know many of us have found that computer games offer a very motivating context for teaching programming, as well as developing pupils' criticality in this medium.

really any different from a 'copy my example' approach to Scratch coding?

I don't think any of us would think that we give young children building blocks to play with so that they become architects or construction workers in later life, although interestingly Froebel himself briefly studied architecture and American architect Frank Lloyd-Wright attended a Froebel kindergarten. Similarly, whilst there will be those we teach who go on to become software developers and computer scientists, this isn't sufficient reason for teaching computing in primary schools; it's more about computing, like music and poetry, being part of a modern liberal education, and about the way the subject contributes to an all-round understanding of the world.

**Occupations:** Alongside the gifts, Froebel identified a number of occupations or activities as part of kindergarten education: these were creative things such as painting, drawing, origami and embroidery. These, of course, still have a part to play in primary education, but for a generation of 'digital natives', the distinction between creative work in a digital, virtual domain and in the analogue, real world is perhaps not as stark as it is for us, their 'digital immigrant' teachers. Is finger painting with Brushes on the iPad *that* different from poster paint on sugar paper? In the days of the old ICT curriculum, we built up a tradition of creative work across a range of digital media in primary education, from 100 Word Challenge blogging through stop motion animation to original compositions in Garage Band. I see no reason to abandon these activities as we move from ICT to Computing. Indeed there's some fertile territory to explore at the digital media edge of Scratch et al, for example turtle graphics, music composition, scripted animation and, of course, game design.

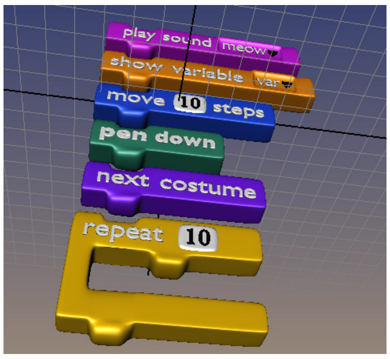
**Play:** Froebel recognised the seriousness of children's play, seeing it as their work. In early years education, practitioners and theorists still see play as fundamental to the learning process; for example EYFS guidance back in '08 included statements such as:

- Play allows children to test their ideas
- Play lets children learn from mistakes
- Play fosters imagination and flexibility of mind

It occurs to me that we could do a search and replace 'Play' with 'Programming' above, and have statements which remain true. Whilst perhaps the work of jobbing programmers in large scale waterfall projects might seem somewhat removed, craft coders, agile hackers and hobbyist makers seem to be engaged in the sort of activities which look, to my naïve eyes, an awful lot like utterly absorbing play.

Just as agile methods place individuals and interactions at the heart of the development process, so a personalised approach to education should place the child, and the child's own enthusiasms, talents and character at the centre.

PRIMARY FOCUS



# REFLECTIONS ON TEACHING COMPUTING AT KEY STAGE 2

**Last September, Hampshire Computing Lead teacher, Phil Bagge embarked on an adventure to start teaching computer science at Key stage 2 in four schools. One year on, he looks back on what he has learnt.**

The challenge at the time was to build a new curriculum to include programming. Programming is challenging and can be hard. Miles Berry aptly described it as 'struggle ware'. In the last six months the learning opportunities I have provided have far exceeded anything taught over the previous 20 years. Seeing pupils rise to the challenge is the reward.

Pupils who enjoy the ordered sequence of a well thought out quiz may not be the same pupils who enjoy creating a racing game. Giving children a variety of different types of programming is important. Let's not get stuck creating endless arcade games or we will lose some pupils' interest. Programming is often seen as a male pursuit so I do go out of my way to try and teach a variety of different types of programming.

Whilst maths may be seen at undergraduate level as an indicator of an aptitude for computing science, things are not quite as clear at primary level. Whether our maths and literacy schemes of work at primary have identified that capacity for logical thinking is open to question. However, a smart teacher will harness their pupils' new-found interest in applied mathematics, whether using decimal fraction to speed up a costume change or Cartesian coordinates to place objects on the screen. Don't be put off by the advanced nature of some of the maths used. Pupils rarely need to understand every aspect to use it. They are adding another layer to their understanding which pays dividends in both disciplines.

We need to free pupils up so that they can make mistakes. Too much ICT is taught whereby the correct outcome is expected first time. Pupils come to believe that they should get everything right first time. The opposite is true in computing science. Making mistakes is totally normal and part of the process of trial, error and debugging. For many pupils this is liberating. When we combine this with the principles of debugging, finding and fixing their own errors, we enable pupils to be far more independent and have positive coping strategies to find and fix failure.

Computer Science A Journey to discover how technology works

Junior omputer cience

forever

Home CS Planning DL Planning Schools Phil Bagge Articles Getting Involved Links Contact Blog Courses

Some of the best learning takes place away from computers. In Year 3 we debug logo code by stepping through shapes on the carpet whilst recording them using a whiteboard. We dance 'Gangnam Style' to help pupils appreciate repeat loops. In Year 4 we design algorithms for early morning routines. In Year 5, write detailed instructions for their robot teacher to create a jam sandwich. Some of this learning is invaluable. Teaching computing science totally without computers would be boring but the judicious use of unplugged time is important. You can find more details of our work at http://code-it.co.uk/index.html.

## FLEXIBLE TEACHING AND DIFFERENTIATION ARE KEY

Programming is very open ended, which is wonderful for pupils but can be scary for teachers who like to stay in control. There will be pupils in your class who have enough time on their hands to learn more than you. We need to embrace this, encourage them to extend their learning even further and harness their expertise to mentor others. This is one area where you are unlikely to be the expert for long. However far pupils extend their programming knowledge they will still need us to build a framework of computational thinking on which they can hang their new found knowledge and understanding.

The degree of differentiation can be enormous. Some pupils will download the software at home and complete tasks outside school. Planning for this is important either by careful questioning that draws out the next step, having extension tasks to hand or new projects they can go on to. I often find by the end of a module of work in Year 5 & 6 that I will have small groups of pupils working on their own projects that have evolved from our common starting point.

It's important not to be too rigid in our definition of those we see as high flyers and those we see as strugglers. As you switch from one type of programming to another, even within the same language, different pupils will shine as the new task grabs their imagination.

Pupils work best where they can collaborate, magpie ideas and re-purpose them. They may be working individually but the importance of sharing ideas informally shouldn't be underestimated. A lot of programming starts with other people's ideas that you use and adapt.

# GETTING TO GRIPS WITH THE NEW NATIONAL CURRICULUM

**John Woollard, Programme Director for the PGCE Primary at Southampton University offers some words of advice for teachers grappling with the change of emphasis implied by the new Computing programme of study.**

The name change to Computing is a simple expression of the curriculum developments but there is nothing simple about the classroom level strategies required to change the pupils' experience from simply 'using computers' to 'understanding how they work'. This article is no blueprint to success but it does describe a strategy for primary phase ICT coordinators to ensure that their teachers move forward with the agenda to introduce computer science to all pupils.

The first in the strategy is, **"do not stop doing ICT!"** That is, enabling the pupils to be both productive and creative through using the computer – those activities continue to be part of the curriculum. Do not stop using ICT to support other subjects such as sensing devices in science, the weather station, digital devices in PE and using the learning platforms to deliver teaching and learning materials – this sort of use is now coined "technology enhanced learning".

Secondly, at this stage, **"do not say to your teachers that this is new or difficult"** – it need not be new or difficult. The National Curriculum for Computing (2014) through its Aims and Subject content provides a rationale for doing many of the activities already embedded in the ICT classroom practice of colleagues. The sidebars in this Primary Focus contain quotes from the National Curriculum for Computing and their exemplification. However, there is a big difference in outcomes – not only are the pupils expected to "do" but they are expected to "understand". With that understanding, the teaching comes with a new set of vocabulary associated with computational thinking; words that will become as familiar as onomatopoeia and phonemes are: algorithm, abstraction, debugging, logical reasoning, decomposing, variables…

The starting point for ICT coordinators could be:

- examine the current long, medium and short term plans and identify activities that can be called computer science, for example, programming a toy to carry out a task;
- re-label those activities on the curriculum computing (CS) - the other computer activities are likely to be Computing (IT);
- identify any gaps in provision by checking the NC document;
- identify any expertise in the school to teach programming. Programming is an effective way of teaching computational thinking;
- ensure that all teachers are familiar with the words of computer science through a short CPD session;
- start introducing activities from CS Unplugged and cs4fn. These do not require pupils to use a computer to complete, they are easily integrated practical classroom activities.

Developing an after-school club can be the starting point for creating a team of expert pupils who can support the whole class in programming activities. If you are initiating a programming activity in your school, the natural starting point is an environment like Scratch. A really good starting point is http://code-it.co.uk/scratch/scratchplan.html.

Teachers work best in collaboration and it is strongly recommended that you join CAS and interact with the CAS forum. Join the Primary forum and don't be reticent about seeking help. Remember, by seeking help others are also helped.

## NATIONAL CURRICULUM: SOME SIMPLE IDEAS

***In 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 a sequence of instructions***

Give them a simple instruction list to walk to the school office using forward x paces; turn rt/lt; etc. The instructions are a form of algorithm. An algorithm is a complete list of clearly defined instructions for completing an activity.

***Write and test simple programs***

After using a programmable toy to follow a route by direct commands, ask the pupils to write down a list of commands before entering them into keypad; if it is not right they change their written 'program' before re-entering the instructions - the pupils are, in effect programming

***Use logical reasoning to predict the behaviour of simple programs***

Here are three simple examples.
Ask, what letter does this draw?

```
Forward 1,
Repeat 4 [Forward 1, Right 90]
End
```

Ask, what shape does this draw?

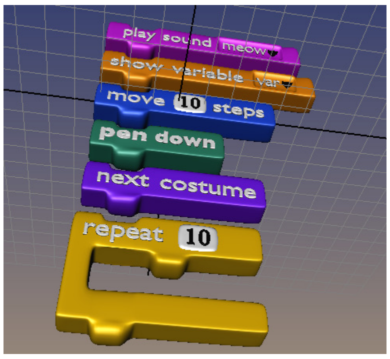
```
Forward 1,
Repeat 4 [Right 90, Forward 1]
End
```

Talk about what these instruction do

```
01- Box A contains 9
02- Box B contains 3
03- Working out
04- Take 1 out of Box A
05- Take 1 out of Box B
06- If Box B is NOT empty THEN
go to 'Working out'
07- How many is in Box A?
```

The pupils could act out the sequence. To help them understand what is happening you could ask What does 9 - 3 = ?

PRIMARY FOCUS



# TEACHING GAME DESIGN IN THE PRIMARY CLASSROOM

**Yasemin Allsop, ICT Co-ordinator at Wilbury Primary School in London shares her experiences of how pupils knowledge and skills are largely shaped by how game design challenges are approached.**

I have been teaching game design to a Year 6 class using the 'Alice' software once a week since January. I have been keeping a journal of my experience to gauge the role of the teacher in the classroom when teaching digital game design. Similarly, some children have also kept a learning journal to record their perspective on their learning. They use screen shots of their problem scenes or codes as a record, then explain how they designed a solution to solve this problem. They also included their feelings and emotions when they were making their digital games.

Some of the children are also in my maths group. This allowed me to monitor if their game design activities impacted on their mathematical thinking. During one session, learning about rotating 3D shapes, I asked the class to explain their strategies for solving this type of problem. One pupil stated that he thinks about the Alice world and visualizes the shape rotating exactly how he rotated the objects on the screen. I started to understand, not only the link between spatial-visual skills and game making, but also how children transferred knowledge from one area to another to solve problems by connecting them. The children saw digital game design as a very similar activity to mathematics, because in their words 'Digital game design was all about problem solving'.

Whilst making games, children transform their mind into a virtual lab where they can develop and test their designs, through thinking (dialogue with 'self' and 'others') and action (dialogue with design) before turning these into reality using software. This is a continual 'making sense' process, where children exercise their planning, decision-making, organizing, testing and evaluating skills; a foundation to learning in many areas. How the game design activities manifests into knowledge is mainly shaped by how it has been taught in practice.

I have been recording not just the children's activities, but also my own thinking of my role as a teacher. I wasn't very familiar with the 'Alice' program, so I was really worried about how to approach teaching it. What I realized is, that by staying in the background and acting as a facilitator, I allowed the children to take part in structuring the lesson in a way more relevant to their needs. In one of the sessions, I used a simple task for designing a quick game on the Internet and modelled it to the class. I wanted to show another one, but the children wanted to be left alone to just get on with it. They wanted to explore Alice further.  So I changed my lesson structure and let them experiment with Alice for the rest of the lesson. They wanted to walk around and see what their peers were working on. Freedom of movement and freedom to discuss with their peers allowed the learning to develop in different nodes. This emphasizes the importance of the teacher understanding their own role in the classroom when teaching game design. I became part of the learning cycle, by learning with the learners. Hopefully I will finish my own game soon, although it is no way near as close to the standards set by some of the children's designs.

## NEW NATIONAL CURRICULUM: SOME MORE SIMPLE IDEAS

Some of the later statements relating to Key stage 1 are probably more familiar to many primary teachers.

***In Key Stage 1 pupils should be taught to organise, store, manipulate and retrieve data in a range of digital formats***
using a word processor and saving work; taking photographs and storing on the computer; recording an MP3 file and saving on a computer; making a presentation

***communicate safely and respectfully online***
use email to exchange messages with friends, experts and automatic systems, talk about being thoughtful and respectful

***keeping personal information private***
talk about profiles such as Whizzkids. Discuss who has access to it and why they should not tell others their password

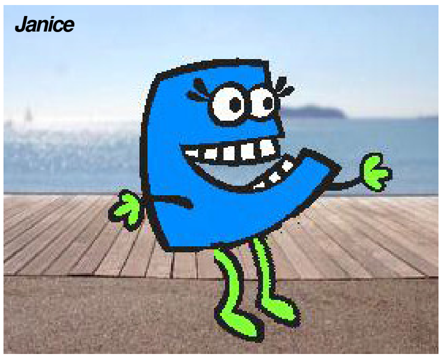
***recognise common uses of information technology beyond school***
discuss with pupils: barcodes on shopping, traffic light systems, CCTV cameras, on-demand television, voice-over-internet. Take a look around your locality, can you find other examples to use? Perhaps you could arrange a 'technology walk'?

{CODE CLUB}

Code Club, one of Google's 2013 RISE Awardees, is taking the UK by storm. Now in over 900 schools and spreading globally with the launch of Code Club World. Google has been supporting Code Club to expand access to coding for 9-11 year olds and have just announced a partnership between themselves, Camden, UCL and Code Club to put a club in every Camden primary school by the end of the year. Let's hope this will be replicated elsewhere.

# WHAT IF THE HOKEY COKEY IS REALLY WHAT IT IS ALL ABOUT?

**What do you get when you mix 34 primary school children, 15 Sense Boards, a robot, an alien called Janice, a tree called Bob, some emergency snacks, the Hokey Cokey, Angry Birds and Julius Caesar? Pete Bell from Rishworth School, West Yorkshire explains.**



Rishworth School hosted a Computing Day for Year 5 pupils from six local primary schools. The pupils visited for two sessions, each lasting two hours, aimed at introducing the fundamentals of Computing: Computational Thinking and Programming concepts. The sessions started with everyone dancing the Hokey Cokey - yes, even the teachers! This introduced pupils to the idea that algorithms are everywhere and that computers use algorithms to carry out instructions. Pupils were then given flowchart symbols and asked to arrange them into an algorithm to explain to an alien (or a computer) how to do the Hokey Cokey. To turn their new-found knowledge of algorithms into a programming task, I had prepared a Scratch template. Pupils used this to create a program that switched the 'costume' of an alien sprite that we named Janice. By pressing a letter on the keyboard the program switched Janice's costume, thus creating movement. By pressing different keys in a particular order, pupils 'instruct' her to dance the Hokey Cokey! All the pupils had Janice dancing in no time.

The Scratch template is shared on the Scratch community here: http://scratch.mit.edu/projects/11002706/. It shows **one** way of accomplishing a dancing Alien. However, **you** could extend this task, too! I'd love to see how you would modify the program and iterate it to do something awesome! Maybe you could sequence the instructions to execute without the need for separate keyboard inputs? Maybe you could create some different costumes for Janice and then program her to do the Harlem Shake? I wish I had some time to do that! Please have a go and hopefully you can share your work and comment on that of others (as is the ethos of the CAS and Scratch Communities).

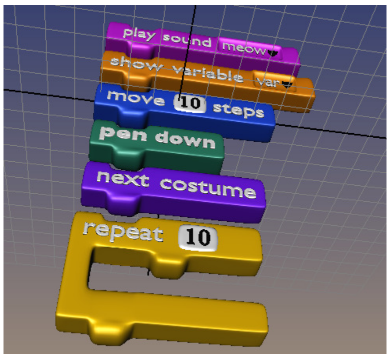
But we had some code breaking to do, so we moved along... I told the pupils that there was someone else in the room who liked to dance the Hokey Cokey. The pupils had to decrypt a message to uncover who it was, using a Caesar Cipher....to hack the code and uncover the secret dancer: a Humanoid robot that I had built and programmed using Lego Mindstorms, over two "gruelling" afternoons.

The pupils then enjoyed some snacks and Bob The Tree oversaw the recycling. Appropriately fuelled with E numbers, the pupils continued their coding endeavours, this time to trigger sound effects and light displays on Sense Boards. As their final task, the pupils debugged a program that monitored a sound sensor, which they then used to control an animation of an exploding Angry Birds character when they made some NOISE!

Throughout, the pupils had been introduced to a range of ideas in the Programme of Study for Computing. In the flowchart task, pupils applied abstraction to decompose a problem into smaller parts. They also applied repetition and sequencing.

During the Cipher task, they used logical reasoning and during the Sensor Board tasks, pupils were introduced to selection, comparison operators and the use of variables to temporarily store changeable values. Of course, given the time constraints, these Computational concepts were not explored explicitly; instead, I took an experiential, constructivist approach.

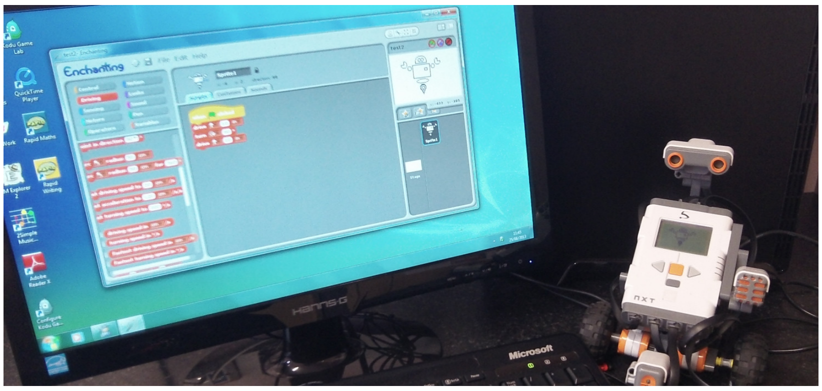
At the end of the first session (and ever the professional), I started my plenary by asking "so, what have you done today?" One pupil simply shouted out: "WE'VE HAD FUUUUNNNN!!!!!" For me, that was mission accomplished.



# ENGAGING YEAR THREE WITH SOME ENCHANTING IDEAS

**At the end of the CAS Conference, attendees were given a postcard to write down one thing that would do on their return to work. Dawn Walker, from Bentley Federation near Walsall wrote: 'make programming real world'.**

Anyone who attended the CAS Conference Primary Forum with Phil Bagge and Jane Waite will know my ideas were shamelessly stolen from them. We had used the Lego NTX robots and Mindstorms software with Year Five and Six pupils over the last 12 months and I was keen to introduce them to the younger children. I discovered a piece of software called 'Enchanting', a variant of Scratch that allows programming of the NTX robots using the familiar, user-friendly 'Scratch' style environment. I was keen to trial it, so when I was asked to work with a Year 3 class on a robot themed cross curricular project, I jumped at the chance.



I would always introduce any robot project with some 'people programming'. This helps pupils grasp key points about algorithms such as accuracy and order. The children are amused when I walk into a wall or keep 'turning' around because a command is wrong! As part of the introduction we also discussed algorithms and learnt a saying/action as a memory prompt. We recently introduced iPads and I was interested to know what value they might have for developing the children's understanding. We used an App called ALEX, similar to the Bee-bot Early Years App. After 15 minutes we started working with the Enchanting software. I set the children the challenge of making their robot move in a square. With some trial and error getting the turning angles correct, most managed it without any help. Pupils then started to explore what we would have to change (and keep the same) in order to map out a rectangle or triangle, so developing confidence applying the techniques to different situations.

By variously applying the programming techniques on humans and iPad / robots, the pupils were able to consolidate understanding and solve the problems posed more intuitively than in previous NTX-based projects we have run. At the end of the lesson I asked the children what they would like to program the robot to do next time. One little boy suggested programming the NTX to take a biscuit to the teacher in the next room. This really made me smile as I couldn't think of anything more 'real world' than that...

## NEW NATIONAL CURRICULUM: SIMPLE IDEAS FOR KS2

Some of the terminology used at Key Stage 2 may be less familiar but they begin to introduce some of the fundamental concepts that the children will be repeatedly exposed to as they progress through school.

***In Key Stage 2 pupils should be taught to: design and write programs that accomplish specific goals***

use Scratch to design an electronic fish tank

write programs in LOGO to draw polygons including stars and combine with RAND to create artistic designs

including controlling or simulating physical systems;

use Commotion control to create an interactive pedestrian crossing

use Flowol to control a 'Ferris' wheel

use Scratch to model a line following vehicle - use of feedback in a control system

***solve problems by decomposing them into smaller parts***

Ask how do we get to school? (break into stages, describe each stage separately)

how do we build a plane from plastic bricks? (describe the stages)

Point out that the children are decomposing complex tasks. A good introduction to decomposition can be found at http://games.thinkingmyself.com

***use sequence, selection, and repetition in programs;***

These constructs lie at the heart of programming. Repeated exposure in lots of contexts is essential.

sequence - a list of instructions

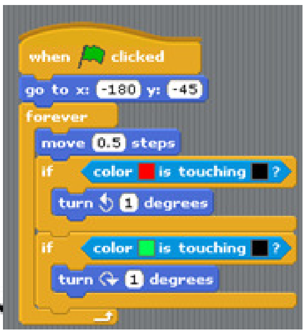
selection - instructions with IF conditions

repetition (iteration) - a list of instructions with parts that repeat

# HOW CAN CHILDREN PROGRAM A ROBOT TO FOLLOW A LINE?

**With 30 years experience teaching Computing and ICT, Graham Hastings (Head of ICT at St John's College School, Cambridge) remains passionate about physical computing because he knows how engaging and enjoyable the children find it.**

This STEM project is designed to teach sensing and control with feedback to children aged 10 to 12. It does not require any specialist knowledge. Physical computing is a perfect context for teaching computing at KS2 and KS3. A common misconception is that hardware is complicated to set up and this is a difficult topic to teach. The project makes use of two motors, switched on if a light sensor receives light and off if the sensor is dark. A black line drawn on a table provides the input by reducing the light level falling on the sensor.



```
when [flag] clicked
go to x: -180 y: -45
forever
    move 0.5 steps
    if color 🟥 is touching ■ ?
        turn ↻ 1 degrees
    if color 🟩 is touching ■ ?
        turn ↺ 1 degrees
```

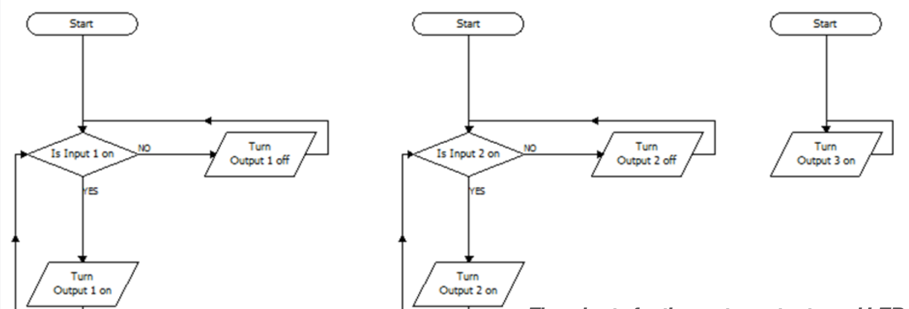
To help understand the theory children can be given a line following problem in Scratch. Once they have solved the problem (above) they should have a good understanding of sensing and control using feedback.

The children can take some time to develop their vehicles, evolving them from manual control to computer control. The starting point is to glue two motors, with wheels attached, at the back of a piece of MDF with a skid (a simple bolt) through the front of the MDF chassis. The motors are wired to the 6v power supply (remove 1 cm of insulation from each, twist together and wrap with insulating tape) and each controlled via a push switch. If the wires are long the vehicle can be steered from a standing position.

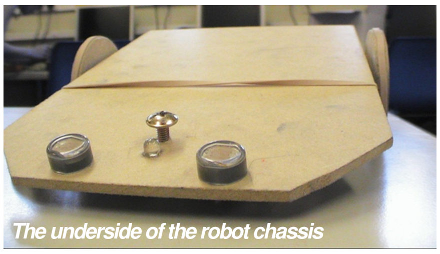
Having experienced two motors being used to steer a vehicle you can discuss the line following problem. How can the vehicle sense a line painted on the floor? It needs to be able to *see* the line. Children can experiment with an LDR, discovering that less light is reflected off a black surface.

The next stage is to glue the two LDRs onto the underside of the chassis, slightly further apart than the width of the black line. A hole can be drilled for the LED to provide the light which enables the vehicle to *see* the line. A resistor is wired in series with the LED to protect it from high current. The final stage of the build is to remove the push switches and wire the components to the FlowGo interface. Connections are made using 4mm spring connectors.

FlowGo Connections:
Left LDR to input 1
Right LDR to input 2
Output 1 to left motor
Output 2 to right motor
Output 3 to LED



*Flowcharts for the motor outputs and LED*



*The underside of the robot chassis*

The control program is set to *Remote* to save it in the interface memory. The interface is mounted on the vehicle and powered by the 6v battery pack. A black line (1 cm wide) is drawn on a light background for contrast. Position the vehicle so that the line runs between the two LDRs and press GO. The bolt not only acts as a skid to allow the vehicle to turn but also provides fine height adjustment. By turning the bolt you can tweak the distance between the line and the LDRs until the vehicle responds. The optimum distance is usually 2mm. Search YouTube for 'Flowgo Vehicle' to watch the vehicle being tested.

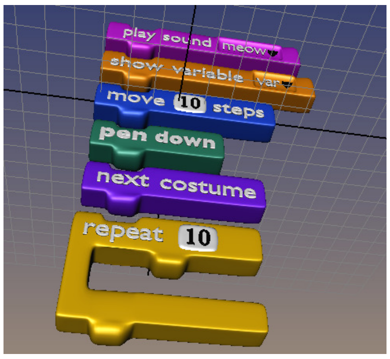
## THE PREREQUISITES OF A GOOD PROJECT

You will need Scratch, Flowol and the FlowGo interface (but you can substitute FlowGo with Arduino, IQ4, Raspberry Pi & PiFace or any suitable micro controller (an increasing number on the market).

Components: x2 3-6v motors with 200:1 gear box, x2 LDRs, x2 push switches, x1 bright white LED, 300 ohm resistor, x10 spring connectors, 2 metres of red and black single ply wire, 6v battery pack (x4 AAA batteries), x2 MDF wheels and a skid (small bolt), a small sheet of 2 mm MDF. Cost about £15 but all can be recycled.

Previously the children should have been introduced to sensing and control through Scratch or Flowol. They should be able to program an output to respond to a specific input. Ideally they will already have come across examples of feedback systems (such as a thermostat) in which the output produces an effect on the input.

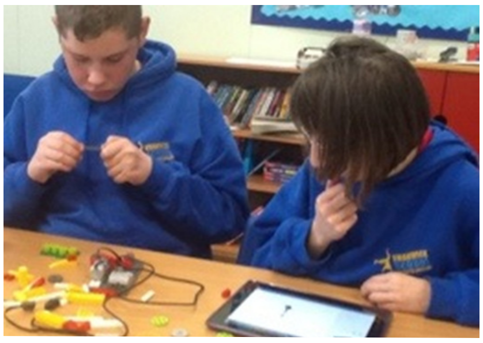
PRIMARY FOCUS



# INTERFACING WITH SCRATCH: LEGO WEDO AND PICOBOARDS

**Matthew Parry is a Primary trained teacher working in a Special School in Derbyshire for students with autism and BESD. A CAS Master Teacher, he reflects on the value of using physical interfaces with programming.**

Using Lego WeDo Construction Kits and Picoboards enables a more kinaesthetic approach to lessons and gives students a more real world reason for programming. Lego WeDo kits contain the usual Lego bricks and four extra parts: a motion sensor (used for proximity), motor, tilt sensor (to determine orientation) and a USB hub to connect the motor and sensors to a computer. Scratch needs no extras to make it work with Lego WeDo. The sensors are automatically enabled but you may need to Enable Motor blocks from the Edit menu to use the motor.



To start I demo a model crocodile with a motor to open and close the mouth and a motion sensor to sense if anything was in the mouth. I then show how to create a simple scratch script that snaps the mouth shut if my finger (or anything else) is placed between the jaws. The students then build their own models using the kits and building instructions which I have downloaded and put on our iPads. Working in pairs facilitates collaboration and encourages sharing of ideas. I do not give them any scripts but try to get them to work out how to get their models to work. They soon get birds moving and tweeting, monkeys drumming and a spinning top humming.

We then attach the Picoboards which allow us to add more control to the models by providing several inputs: a sound sensor, light sensor, push button, slider (potentiometer) and four resistance measuring inputs. After installing the necessary drivers, Scratch 1.4 automatically enables all of the sensors and the students soon work out how to control their models by clapping or pressing the button. Some use the slider to control the speed of the WeDo motor; some build fancy mechanisms to control the amount of light and hence the volume of sounds they are playing. Other students investigate resistance of everyday objects by connecting all sorts of things to the resistance inputs to see what works best.

The kits are quite expensive, almost £80 for a WeDo kit, but as they use standard Lego bricks they are extendable and are accessible for all ages from infants to seniors. The Picoboard (£35) is not limited to being used with WeDo kits but can be used to interface with any Scratch program. Using the kits has allowed my students to be creative and to explore areas of Scratch they would otherwise not experience if they only used the computers. I try to not be prescriptive in what I want them to achieve allowing them instead to discover what they can do and share with each other their results. The only limit is the students' imaginations. For CAS members, links to helpful resources can be found on the Community at http://community.computingatschool.org.uk/resources/1211

## NEW NATIONAL CURRICULUM: MORE IDEAS TO USE IN KS2

The key areas in the new KS2 programme of study relate to programming. Some terms may be a little unfamiliar. Here are some easy to implement interpretations

***In Key Stage 2 pupils should be taught to: work with variables***
create a Scratch game that keeps the player's score

***and various forms of input and output***
generate appropriate inputs and predicted outputs to test programs
simple escape the maze activity in LOGO - load an image of a maze with the turtle at the centre - children have to plan a sequence of instructions that will steer the turtle out of the maze.

***use logical reasoning to explain how a simple algorithm works and to detect and correct errors in algorithms and programs***
See the introduction to algorithms at http://games.thinkingmyself.com

A := B means 'make the contents of A become the same as the contents of B'
A := A + B means 'make the contents of A become the same as the contents of A added to the contents of B'
A := 7 means 'make the contents of A become 7'
A can be a box (of counters on the desk), cell (of a spreadsheet) or memory location (in a computer). Using the following:

```
A := 5
B := 7
A := A + B
B := A - B
A := A - B
```

What is the content of A now?
What is the content of B now?
What does the program do?

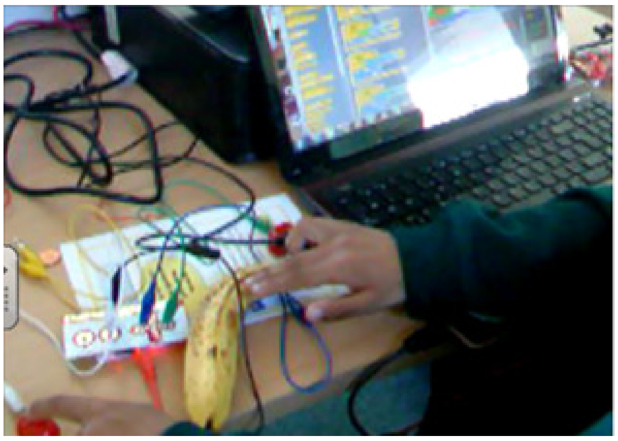
# COMPUTING DAYS HELP FORGE PRIMARY / SECONDARY LINKS

**Inviting primary schools to a computing event can be a source of inspiration for all involved. Sue Gray, who teaches at Fakenham High School in Norfolk, reports on a couple of very successful intensive days they organized for local primary school children.**

In June a group of primary pupils visited us for a fun day of Maths and ICT. We began with *Stompy Zombie Robots* in the Drama studio! Two teams programming 'real' (pupil) robots to fire tissue paper missiles. Then off to the ICT room to learn some control with Flowol. Pupils created flowcharts to control flashing lights and use inputs such as sensors. Finally the main event – learning to program the Lego Mindstorms robots. But first, some maths to do – gathering data on how far robots run on 50 or 75% power for 5 or 2 seconds etc. Entered into a spreadsheet, we could then analyse the data. Various challenges built on the data gathered. First the robots had to move between two floor crosses. The next involved a 90 degree turn. Finally came the maze, each team programming the robot to run through without touching the sides – tricky stuff but one group did an *a-maze-ing* job! Our winners were fantastic, with only a slightly dodgy 3rd turn, they ended almost exactly on the finish X!

The pupils were put into random groups so developing the team spirit as soon as possible was essential. They were great at working together. It was a fabulous day and thoroughly enjoyed by pupils and staff alike.

In July a smaller G & T primary group joined us for an intense two day experience. We planned to teach them a little Scratch, set them loose with a couple of worksheets and then give them a challenge. As it turned out a couple of them had already experienced Scratch through an after-school Code Club. These pupils were keen to demonstrate their skills. Others had no experience at all but quickly picked up enough skills to create a very simple platform game.



We also had the opportunity to play with Makey Makey and the Picoboard – both were judged to be "awesome". Using the Makey Makey board we wired up some cherry tomatoes and a banana (my lunch!) and used them to control a character. The Picoboard did a similar thing using a slider control, the button and sound sensor. This was fantastic as they could shout at the character to make it jump!

Once we'd made our games we moved on to marketing – a bit like The Apprentice! On the second day we created an insert for a CD case and a poster using our game colours and some of the lettering from Scratch. Finally, pupils had to write a script for a 25 second radio advert which proved quite hard. All left with a USB containing their Scratch game, pdf files of their CD insert and poster as well as media files of their advert. All the pupils thoroughly enjoyed their visit. The only real downside to the two days was that there were no girls. Next year we will try to encourage the primary schools to put more girls forward for these intensive days.

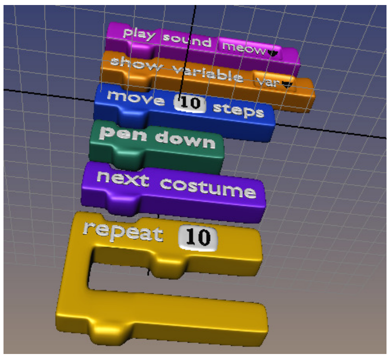
## CAS COMMUNITY LETS US SHARE RESOURCES

I am a Year 1 teacher at West Hove Infant School in Sussex. We are a large 8 form infant school. I am always keen to acquire great ideas and resources from other teachers that I can easily share with my colleagues. The CAS Community Resources allows teachers to post, share and re-work resources. I had been trawling the CAS forums for ideas and resources to help us explain the web to our little 5 year old pupils and came across a fantastic KS2 Powerpoint 'Modelling the WWW' from Graham Hastings (community.computingatschool.org.uk/resources/178).

I altered it only slightly for KS1. The presentation gives you all the instructions to undertake an unplugged lesson with children role playing Webservers around the world, Google and a Researcher. The children found it hilarious that they got answers back from 'Who wrote Winnie the Pooh' as a Pooh Holiday Park; a picture of a Polar Bear and a lady with a nickname of 'Winnie'. It was a great lesson as it started to dawn on them that maybe the web was not as 'clever' as they thought and maybe it didn't always tell them what they needed.

We followed this up with the super Netsafe Utah video on "What is the Internet" http://www.netsafeutah.org/kids/kids_videos.html. Then I went on one of our learning platform blogs and we posted something as a class. Suddenly lots of them were asking, so we could say anything on the web, and anyone might put anything and they could be anywhere. It was a light bulb lesson and we keep referring back to it when ever we are looking at issues to do with the internet. It's a great lesson. I'd recommend it for KS1 or KS2! *Jane Waite*

PRIMARY FOCUS



## NEW NATIONAL CURRICULUM: MORE IDEAS TO USE IN KS2

***In Key Stage 2 pupils should be taught to: understand computer networks including the internet;***
'draw the internet' activity following some descriptive input/experience http://www.canyoudrawtheinternet.com/uk to develop a collective understanding - the teacher should identify and correct misunderstandings/misconceptions

***how they can provide multiple services, such as the world-wide web;***
The internet is used to carry information via web pages, email, voice messages, file transfer and many other means using protocols such as http, ftp and smtp. https is a secure way of sending information through web pages; it is used by banks and shopping web sites.

***and the opportunities they offer for communication and collaboration***
Here are some of the ways that the internet offers means for communication: Wikis, forums, e-portfolios, blogs, chat rooms, voip and social networking. How many have your pupils used?

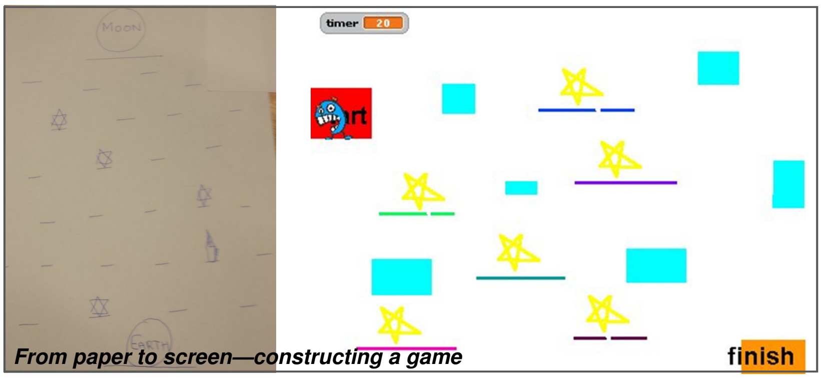
***describe how internet search engines find and store data;***
A search engine uses a program called a spider to visit web pages and put them into lists according to their content. The pages are ranked by popularity. Pages with the most links to them or the most used are ranked higher. When you use keywords in a search engine, the web page displays the links from the most appropriate list.

***use search engines effectively;***
this requires the use of appropriate keywords (what are good words) and being discerning in evaluating digital content.

# PRIMARY PUPILS ENJOY A FULL DAYS MINI GAME JAM

**For three years Amanda Wilson's PhD research at University of the West of Scotland has focused on Games-based learning in Upper Primary Education, working with local schools and using Scratch to construct games.**

As this year was the last year of my data collection I thought about an event to give something back to the schools and also show the skills children have picked up while being part of the research. The Scottish Game Jam (an intense weekend of game design) inspired me to organise a one day game jam of my own for children. After lots of emails and preparation the Mini Game Jam took place in June. 52 children from Royston and Carntyne Primary in Glasgow (mainly primary 7's but a few primary 6's) attended the University of the West of Scotland to participate. The day started with the children being put into teams for the day which meant working with children from the other school – I would have been more popular saying they had to work with aliens for the day! That aside, the teams worked well and given most are headed to the same secondary school it was a great transition exercise too.



*From paper to screen—constructing a game*

During the day the children came up with lots of great ideas from quizzes to football games to maze games. Later they were transferred from paper to the screen. By the time 3pm came most of the groups had managed a game (some more complete than others) and some even had time to do some play testing to ensure things were working. While children enjoyed a snack the games were judged, the winner picked for its originality. Their game, Nector Collector, involved controlling a bee to collect nector (coloured circles) while avoiding black blocks in a fixed time. The game can be played online at http://scratch.mit.edu/users/mini-game-jam/ along with some of the other games created on the day.

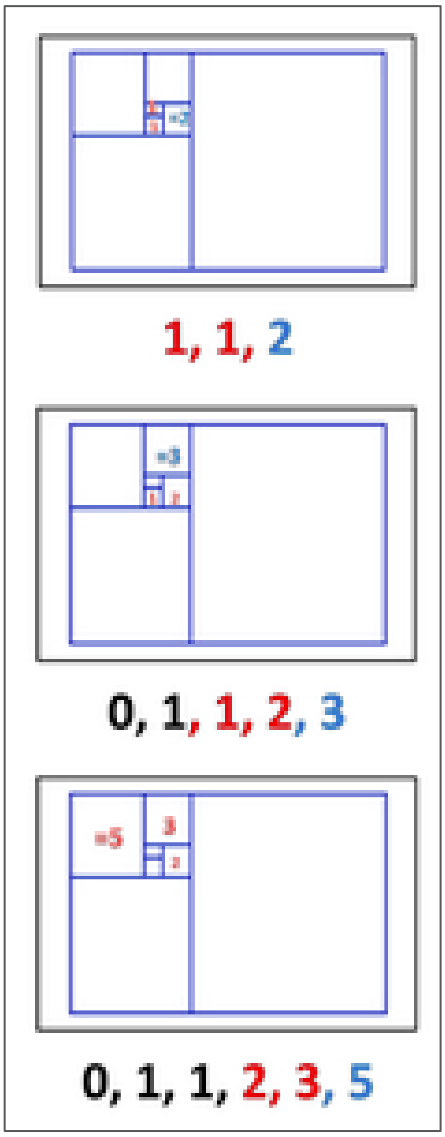
I was lucky enough to have Dr Daniel Livingstone as a volunteer and he kindly gave the children a talk about the games industry in Scotland and where graduates go once they leave UWS. It was a very informative talk for the children and hopefully gave them all something to think about.
I would also like to thank the following people without whom the event would not have been possible. Prof Thomas Connolly – for letting me go ahead with my idea in the first place. Simon Kelly and Jo Church – Heads of both schools for their help along the way and letting their schools participate. Dr Thomas Hainey, Dr Jon Sykes, David Moffat and Maxine Dodds for coming along and helping out. Finally the School of Computing UWS, Computing At School, Computing At School Scotland and Science Connects for all their help with sponsorship of the event and prizes too.

# EXPLORING ALGEBRA, ART AND FIBONACCI THROUGH COMPUTING

**Langley Grammar School's Digital School House (DSH) project has worked with their Maths and Art departments to combine the teaching of algebra with the Golden Ratio. Mark Dorling explains how maths, art and computing can combine.**

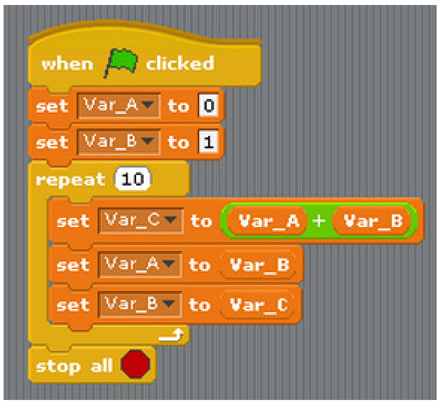
This lesson has been designed to make algebra more relevant to pupils by applying it to real life scenarios whilst teaching computational thinking concepts. It is a lesson that really inspires the pupils by helping them to see the relationships between Computer Science and the other subjects they study! It also provides an excellent opportunity to teach the basics of and give pupils of all ages an introduction to recursion.

The idea for this lesson came from the Computer Science For Fun (CS4FN) website which highlights the relationship between Fibonacci Series and the Golden Ratio which is evident in both nature and man-made objects throughout the ages e.g. pine cones and ancient greek architecture. The school's Art department teach the introduction to the Golden Ratio and relate it to their curriculum through the study of portraits. They also use the Internet searching and referencing skills that have been previous developed in IT lessons to undertake research projects into where the Golden Ratio occurs.



The Digital School House teacher begins the lesson by investigating what the Golden Ratio is and where it appears in nature. Pupils then develop their artistic skills by drawing portraits of one another using the Golden Ratio to help them. Using the Golden Ratio pupils are then introduced to the Fibonacci Series and they are supported and encouraged to spot the pattern i.e. the new number is equal to the previous two numbers.

If the pupils have not covered what variables are then the DSH teacher introduces them to the concept by completing some simple mental maths where the missing number floats between the two numbers to be added and the answer. Pupils record the answers on personal whiteboard – drawing a flow diagram to represent the calculation. The purpose of doing this is so that the pupils can analyse the structure of the equations i.e. number 'a', followed by the 'operator' and then number 'b' is equal to 'c' i.e. 'c' = 'a' + 'b'. The teacher uses this introduction to algorithms by getting the pupils to make a simple calculator in Scratch (using their flow diagram as guidance) enabling the teacher to cov-



```
when [flag] clicked
set Var_A to 0
set Var_B to 1
repeat 10
    set Var_C to (Var_A + Var_B)
    set Var_A to Var_B
    set Var_B to Var_C
stop all
```

er the difference between 'constants' and 'variables'. It is essential to do this activity before progressing on to solving the Fibonacci Series. Pupils are then supported to model (using a flow diagram) the pattern they spotted previously in the lesson (Fibonacci Series) and now that they know what a variable is and what it does, they are able to further develop their flow diagram to include the movement of values between variables. Pupils are then introduced to iteration (repetition) with the purpose of growing the Fibonacci Series. Finally, pupils are encouraged to further develop their Scratch program (above) by modelling the Fibonacci Series using their flow diagram to guide them. You can download the resources from http://tiny.cc/tps70w

## FREE RESOURCES FOR THE NEW CURRICULUM

In January Primary schools across the country will, courtesy of Microsoft and Rising Stars, be receiving a free starter pack of 'Switched on Computing' resources to help them get started with the new curriculum for computing. The resources will be full of engaging and fresh ideas that bring the programme of study alive, with support for teachers who need a helping hand.

The resources are a taster for the bigger Switched on Computing scheme also published by Rising Stars in January. Teachers can pick up free copies at BETT and digital versions will also be available on the Rising Stars and Microsoft site.

# DESIGN BRIEF: AS IT'S BROKE, LET'S FIX IT REFORMING TECHNOLOGICAL EDUCATION

**The STEM strategy has not had the desired effect on the skills shortages threatening UK economic competitiveness. We need a solution which can be implemented speedily, cheaply and with as little disruption as possible, argue Adrian Oldknow and Tony Houghton from The Centre for Innovation in Technological Education.**

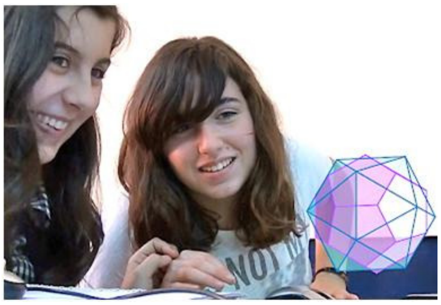
## A BIT OF BACKGROUND

In 2001, Sir Gareth Roberts reviewed the supply of science and engineering skills in the UK. `SET for Success' was published in 2003. It set out 37 recommendations to develop the supply of people with STEM skills into research and industry. All were accepted, including the establishment of a national and nine regional Science Learning Centres. It also established the Science Council, alongside the Engineering Council, and effectively recognised mathematics as a science. However, the strategy which followed concentrated mainly on enrichment in science. It paid little attention to technology, engineering or computing. There is a need to establish coherent cross-curricular activities to encourage the personal, learning and thinking skills demanded by employers and HE.

The CBI and other employers' federations have regularly reported on the effects of skills shortages on key UK business, such as aerospace. NESTA published its `Next Gen.' report in 2011 pointing to the severe skills shortage in the video, games, animations and special effects aspects of creative industries. The `Next Gen Skills Campaign', backed by Google, Microsoft, BCS and e-skills, played a key part in ICT being replaced by Computing. It received a significant boost when Eric Schmidt (Google) noted: *"Over the past century, the UK has stopped nurturing its polymaths. You need to bring art and science back together."*

The DfE have now published the outcome of their curriculum review. It is clear that schools will have considerable adjustments to make from September 2014 so now is a good time to encourage a broad review of technological education.

A group of school leaders and STEM subject associations (ASCL, ASE, CAS, DATA, MA, NSEAD and Primary Engineer) developed the SySTEMiC strategy to embed STEM in the curriculum. It was very well received, but no organisation was forthcoming to take it on board so we decided to do it ourselves, establishing the Centre for Innovation in Technological Education in Cambridge (CCITE – http://ccite.org). ASCL and CBI are supporting CCITE's development of `Technological Education For All Pupils'. By the end of 2014 it will have assembled and tested a complete know-how kit for schools on how to deliver and manage a whole-school approach to technological education. This will support schools delivering the new D&T and Computing curricula as well as enhancing maths and science. It will also develop pupils' personal learning and thinking skills in aspects such as problem-solving, team work and communication, much sought after by employers. A key component is resources for 20 half-termly, cross-curricular, problem-solving projects in both Key Stages 2 and 3 – the `20-20 CCITE STEM projects'.



Students apply maths, science and computing skills to design solutions to challenging problems and make artefacts. They will build up an accredited portfolio of work, and will be in a position to make well-informed choices about future subject choices. They will also be better equipped to see the importance of curriculum subjects such as science, D&T, computing and mathematics and better motivated to achieve good results. Schools will receive guidance and support on a variety of ways to engage talented members from a wider community including parents, governors, older students, employers and employees.

So let's get started. Putting the UK's technological education on a secure footing is a vital and challenging task. We believe we have come up with the principles to tackle this successfully within the constraints of time, money and disruption. We estimate the costs to flesh these out, test, modify and disseminate to be around £2m over 2 years. It is important for the financial stakeholders to be drawn widely—mainly from business and industry, but with opportunities for charities, philanthropists, government and others to participate. We just need to find around 40 contributors to take out shares to raise the capital required. As well as help in raising funds, we need participation of experts from many fields to flesh out the detail – particularly the content of the 20-20 projects. We also need early adaptor schools, colleges and academies (5-19) to test out and help develop the materials.

You can contact Dr Tony Houghton – CCITE Education Development Director on ajh249@gmail.com and Prof Adrian Oldknow – CCITE Founder via adrian@ccite.org

# THE FIFTH ANNUAL CAS TEACHER CONFERENCE THE BIGGEST YET

**"This was my first time at the CAS Conference for Teachers and the energy and excitement I felt was amazing – there was a real buzz" reports Liz Hadley, the Head of ICT at Kenilworth School, West Midlands.**

From arriving at University station on the Thursday evening until the final plenary session, I was welcomed by smiling student ambassadors, provided with ample food and drink, and had queries dealt with efficiently and helpfully. At the Thursday evening reception, hosted by OCR, we were offered wine, soft drinks and a tasty buffet – and more wine (so glad I wasn't driving!). This was followed by an excellent talk and demonstration of "blockly" by Neil Fraser, and then a "Dimbleby-style" *Question Time* session which sparked some interesting debate.

Having registered and signed up for the workshops (of which there was a vast selection) the previous evening meant a respectable 9.00am start on Friday. The first talk, by Michael Kölling, introduced ways to motivate students through programming. There was a collective "Wow!" when Michael demonstrated how the Xbox Kinect can be hooked up to Greenfoot to create games where the player actually appears on the screen – just like something out of *The Matrix*. Next came a demonstration, by Lee Stott and David Renton, of how Microsoft's TouchDevelop can be used to create an inspiring games development curriculum. Having played with TouchDevelop I hope to use it in my classroom soon.

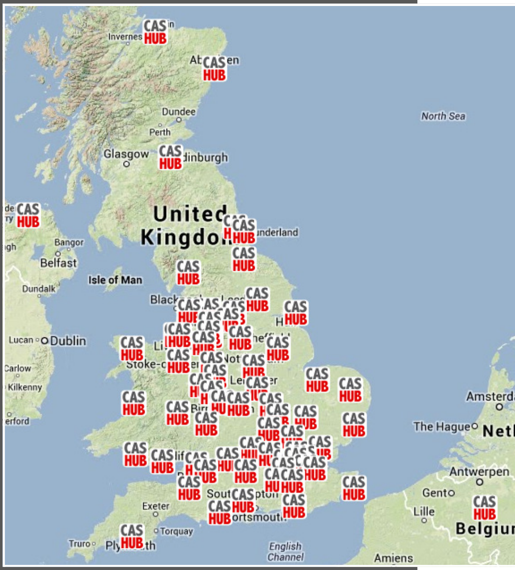
After a refreshment break delegates were then dispersed to their chosen workshops. I chose sessions with a focus on techniques for teaching computational thinking, engaging and inspiring students and building an outstanding curriculum. Looking back on these sessions some common themes emerged:

- You can teach computational thinking away from the computer (*CSUnplugged* has great resources)
- Relate programming exercises to real life problems that interest your students
- Hacking (modifying existing code) can be an effective teaching tool

One of the great things about attending the CAS conference was the opportunity to meet and chat to other teachers and IT professionals. It was reassuring to discover that my school is not the only one where more time has been allocated to core subjects, languages and humanities, leaving little time for computing in Key Stage 3. I was encouraged by the experiences of others and by the sheer enthusiasm that so many at the conference have for our subject. I felt very much at home amongst members of my local hub and colleagues from similar schools. I came away feeling that I belong to a fantastic community of like-minded professionals who really want to inspire and excite young people about computing. Main Highlight? Alan O'Donohoe's session – need I say more!

## POPULARITY GROWING AS CAS HUBS DOUBLE

There are now 70 registered CAS Hubs, a 100% increase since last year, with our first steps into Europe with the CAS Western Europe Hub being launched in Brussels! A warm welcome to new Hub groups recently set up in Kirklees, Stockport, Wirral, East Surrey, Sunderland, Edinburgh and North Kent. Also there are three dedicated Hubs aimed at offering computing support to Primary audience – East Suffolk, Southampton and Stockport. Phil Bagge is leading the Primary Groups.



There are still areas of the country which have no hubs within reasonable travelling distance, so you may want to volunteer to set up your own. Experience shows that the local Hubs are the main mechanism through which to encourage collaboration and development at a local level. To find your nearest Hub, check the CAS Community interactive map or your CAS Community profile will inform you of the nearest one. *Claire Davenport*

Applications for the next round of Google RISE Awards will close on 30th September. Details at http://www.google.com/edu/programs/google-rise-awards. RISE Awards are designed to promote access to Computer Science education and give students aged 9 to 18 the opportunity to become creators - and not just consumers - of tomorrow's technological innovations. We award funding to, and partner with, organizations running programs that inspire, engage, and retain students on a long-term path in the field. Organizations will be chosen for their track record of success, focus on underrepresented minorities in the field, potential to scale, and positions as changemakers in their respective countries for Computer Science at the Pre-University level. *Peter Dickman*

# NOW WE ARE 6

**CAS has trebled in size in the last year, and trebled in size the year before. The majority of CAS members have joined in the last couple of years and might not be aware of where we came from. As our sixth anniversary approaches, SwitchedOn provides a quick history lesson.**

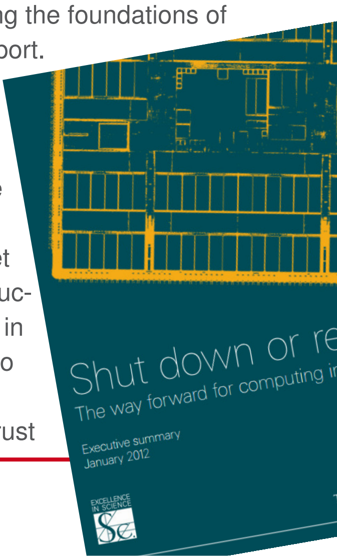
In early 2008 a small group of concerned individuals started to discuss the lack of computing in schools. CAS was born, but none could have imagined back then that within six years Computing would be part of the National Curriculum. Indeed, the first tasks facing this small group was to get anyone to listen! As the driving force and Chair of the group, Simon Peyton-Jones, explains, "It was rather like being at the bottom of a deep well, looking upward and shouting 'Computer Science is important!'".

Although small, the initial meetings drew individuals from a wide variety of backgrounds including teachers, academics, parents, representatives from industry and examining bodies. The lack of hierarchy and broad composition continues to have a lasting impact. CAS is fundamentally a grassroots organization, interested primarily in 'walking the walk'. If something needs doing, people step forward and do it.

From the start, it was clear that CAS had touched a nerve. Many individuals felt, for various different reasons, that things had gone wrong. In the high tech industries, areas of research were expanding but there was a shortage of capable graduates. At undergraduate level, the intake for Computer Science degrees had fallen to worrying levels, and within schools there was a growing concern about the type of course that had come to dominate KS4 provision, which subsequently shaped what was taught lower down. But there was also a thirst to rediscover some of the concepts and pedagogy that had driven earlier curricula. Initiatives like cs4fn (Queen Mary College, London) and CS Inside (Glasgow University) had a small but appreciative audience amongst teachers. The first CAS Teacher Conference took place in June 2009, drawing many of these individuals together, with a keynote address from Tim Bell who had pioneered the hugely successful CS Unplugged initiative.

Over the next five years, CAS transformed itself, from a small 'guerilla group' into the subject association for Computer Science teachers. Its activities diversified. If CAS wanted Computer Science to be embraced in schools it needed, in conjunction with others, to make the case at a national level. Several strategically placed CAS Board members have worked hard to gain the ear of the DfE and many have contributed submissions to various reports and consultations. The most significant was the18 month study by The Royal Society which culminated in the influential report 'Shut Down or Restart?'. Published in January 2012 it marked a turning point. Only 18 remarkable months later, in September 2013; the DfE published the new National Curriculum for Computing that establishes computer science as a core subject discipline, from primary school onwards, one of CAS's key goals.

As well as lobbying for change CAS were busy building the foundations of a network of support. The number of local hubs grew quickly. These have become the bedrock of CAS, over 70 now meet regularly. Their success, particularly in developing face to face contact and building mutual trust



NETWORK OF EXCELLENCE COMPUTER SCIENCE TEACHING

A year ago this month the **Network of Teaching Excellence in Computer Science** (NoE) launched. CAS, working in collaboration with the BCS Academy co-ordinates and provides training opportunities for both existing teachers and those training for the profession. The DfE have supported the application made by CAS/BCS to continue and expand the NoE. The heart of the programme is to build a high-quality, sustainable CPD infrastructure at low cost. This will be achieved by nurturing long-term, bottom-up collaboration between employers, universities, professional bodies, schools and teachers. In the first six months of the scheme 622 schools and 70 universities committed to being involved. 120 schools self-designated as lead schools and 28 CAS Master Teachers were recruited and ran over 700 hours of local CPD events. All the evidence so far points to this being a model that is valued by teachers.

The new funding is for two years but we are working on a five year programme to:

- Recruit and train 600 Master Teachers (primary and secondary)
- Harness university expertise to train and develop Master teachers
- Maintain classroom resources for all key stages
- Enhance professional status for all Heads of Computing in schools

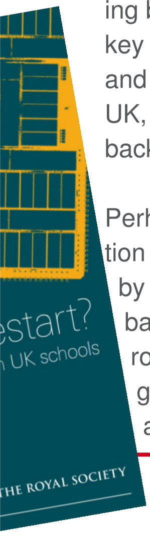
We've made a start already but time is short. If you haven't already done so, please **register your school as part of the Network**. By doing this your school is saying that Computing is important. **Register your school as a Lead School** to support colleagues in other schools and consider **applying to be a CAS Master Teacher**. Don't be shy. Remember, there is no them, only us!

*The CAS Board is chaired by Simon Peyton-Jones (right).*

*Left to right, top row; Bill Mitchell, Simon Humphreys, Shahneila Saeed, Thomas Ng, Michael Kolling and John Woollard. Middle row; Kevin Bond, Kate Farrell, Peter Dickman, Mark Clarkson, Mark Dorling and Pete Bradshaw. Bottom row; Clare Riley, Miles Berry, Tom Crick, Roger Davies, John Stout and Stephen Hunt.*

It might seem strange to those who are new to CAS, particularly given the multitude of events that are now being organized, but the structure of CAS is very simple, and small. The CAS Board (above) oversees the day to day running. Formally, CAS is part of the BCS Academy who provide welcome financial and administrative support. Bill Mitchell, Director of the BCS Academy of Computing has worked tirelessly lobbying on behalf of CAS. Two other significant sponsors are Microsoft and Google, represented on the board by Clare Riley and Peter Dickman. Iain Phillips represents the Council of Professors and Heads of Computing and other key contributors from University Computer Science departments are Michael Kölling, developer of Greenfoot (University of Kent), Stephen Hunt (University of Hertfordshire) and Tom Crick (Cardiff Metropolitan). Teachers include Shahneila Saeed (Graveney School, London), Mark Clarkson (Egglescliffe, Cleveland), Roger Davies (QES, Cumbria) and John Stout (King George V College, Merseyside). Former teachers Simon Humphreys (national co-ordinator) and Mark Dorling (CPD co-ordinator) work full time for CAS. John Woollard (University of Southampton), Pete Bradshaw (Open University) and Miles Berry (Roehampton) are all involved in teacher education. Kevin Bond is Chair of Examiners (AQA Computing) and Thomas Ng works for West Berkshire's School Improvement Team. Through this group we try to maintain contact across all interested parties. The board meets twice a year at open working group meetings to which other active members are encouraged to come along and get involved.

has had a profound impact on shaping the ethos of the developing Network of Excellence. The enthusiasm embodied in CAS is rooted in practice. It is no talking shop, but a body that does things, with few resources, little funding but an abundance of energy. The key challenge now is to win the hearts and minds of ICT teachers across the UK, many of whom do not have a background in Computer Science.



Perhaps the most significant contribution to this is CAS Online. Developed by Michael Kölling and Neil Brown, based on their successful Greenroom Community, it provides a fast growing resource repository and active forum. CAS Online's membership trebled in the year 2011/12, and trebled again in 2012/13; we now have close on 6,000 members.

We want to encourage new teachers to raise questions. There is an unparalleled generosity of spirit amongst CAS members. Those outside teaching want to help those at the cutting edge. We need teachers to tell us what the problems are and point the way to solutions. We want to provide a welcoming environment where no question is too small or simple. CAS doesn't have all the answers. Humility is more important than subject expertise. We have a once in a lifetime chance to shape our subject. Just like children, we will learn through what we do.

## LEADING COMPUTING IN SECONDARY SCHOOLS

The National STEM Centre in association with CAS and NAACE is offering intensive professional development in leading a computing department to new and aspiring heads of Computing/ICT in secondary schools. It will cover leadership, monitoring, coaching and developing team members through six days of residential development and three on-line gap tasks. It will be set in the context of the substantial curriculum changes that are happening in the subject area of Computing.

### FIRST RESIDENTIAL

**Day One:** Creating a Vision for you and your Department. How did we get here? Identifying the challenges and the opportunities.
**Day Two:** Computational thinking, computer science and computing. The pedagogical and infrastructural challenges of the new curriculum. Assessing learning.
**Day Three:** Developing outstanding teaching and learning. A Computing team rather than a Computing department.

Gap Tasks look at understanding your personality type, identifying where are you now and recognising infrastructure challenges.

### SECOND RESIDENTIAL

**Day One:** Using Data, Action Planning, Monitoring and Evaluation. Getting the CPD required.
**Day Two:** Project work. Creative and cross-curricular work. Digital literacy. Design cycle.
**Day Three:** Coaching. Developing momentum and delivering the vision for your department.

The residential programme will be held at the National STEM Centre, University of York. It will run in Autumn 2013 and Spring 2014. For further information go to: http:/www.nationalstemcentre.org.uk/computing *Paul Browning*

# HOW DO YOU TEACH PROGRAMMING? YOU CAN'T DO EVERYTHING AT ONCE

**A key aspect of Computer Science is programming. Teaching young pupils how to program is not an easy task but teachers are definitely up to it argues Chris Roffey. The key is not to try to cover it all at once. Chris teaches at Ewell Castle School in Surrey and is author of the Coding Club series of books.**

## TEACHING IS DIFFICULT BUT TEACHERS ARE GOOD AT IT

Please indulge me while I explain that teaching is difficult but UK teachers are good at it. I feel the need to do this because no matter how well intentioned, it is easy for those outside the profession to believe that anyone who has brought up kids, has been to school themselves or run a club can do it—an idea often pedalled by the media and politicians.

Teachers have the job of enthusing, inspiring and encouraging 30 children at a time to work hard on subjects they may find difficult, have no natural interest in and are under a compulsion to study. The lessons must show clear aims, include formative and summative assessment and pupils must make demonstrable progress. Subsequent lessons must be planned based on the outcomes of these assessments.

And yet, teachers are judged against that one inspirational teacher that everyone had when they were young. They are judged to have failed if their lessons are not outstanding. Teachers' ability, like every other trained professional, is bell-shaped. Even average teachers, those of us that have more than the occasional bad day, are doing a remarkable job. That grades have improved over the last 30 years is a measure of that. Teachers constantly improve their strategies to help children through new hoops and new specifications. UK teachers are good at teaching.

How to teach programming… It might seem an overwhelming task: Which language to teach? What languages should students learn? How can we expect ICT teachers without programming experience to teach sound coding principles? A simple suggestion follows, but one thing is certain, teachers will rise to the challenge

How to teach programming… Although it might seem overwhelming this article really makes only one point: It is very important that we do not feel the need to teach everything at once. Seems obvious, but just wait until you start discussing with programmers what to leave out. To get this right requires mutual respect between all in CAS and for teachers to value their own skills. I would suggest, for example, that in year 7, teachers choose to only teach 'while loops'. "But they are not appropriate for iterating through arrays!" some will scream. Then don't teach arrays in year 7! There is plenty of fun to be had with 'while loops' and simple functions, but more importantly the students can become experts in them before reaching year 8. Then they will appreciate how much more useful 'for loops' are in certain circumstances. Most importantly they will not become overwhelmed and turned off in year 7.

I have found gifted year 6 students can cope well with one sort of loop, writing functions and simple variables but when presented with arrays they start to moan: "This is really hard sir." On the other hand average students in year 8 (progressing through the same material at a similar pace) do not hit this wall. Other teachers I have spoken to report similar experiences. Most pupils at year 6 do not appear to have the cognitive tools. Arrays are more difficult to learn than simple integer and string variables. Consequently we must not ask children to move on too soon as this will only confuse them. **We must give time for consolidation.** My experience is that children that have tried to jump to arrays and stumbled have still enjoyed going back to carefully selected projects which allow them to consolidate and use, in new ways, what they have learnt before but which is not yet secure. When they are ready, they will then be able to cope with more advanced material.

I suggest educationalists resist the advice of those who say we cannot leave anything out; who say that some children will never be able to code; who propose that the best way to learn is for children to do it themselves. Like any subject, programming can be introduced to everyone in carefully structured manageable parts. The reason some people "do not get programming" is because they do not have the necessary foundations and are expected to suddenly learn everything. This is not how Maths or Science is taught. Teachers of Maths and Science have an unfair advantage though, they have carefully planned teaching schemes that have evolved over a long time, time that has not been available to Computer Science teachers.

If a carefully planned programme was introduced nationally teachers without a computing background could more easily be trained. Teachers could become competent exponents at certain levels without worrying about being expert programmers overnight. In other words we could expect teachers to train themselves up in carefully structured chunks with time to consolidate their skills, just like their students. This is another thing UK teachers are really good at.

# SGD: GOING BEYOND DRAG AND DROP VISUAL PROGRAMMING

**In the USA the Scalable Game Design (SGD) Initiative has become the largest National Science Foundation funded study of game/simulation design at middle school level. Principle Investigator, Alex Reppening from the University of Colorado explains.**

Incredible momentum has been building internationally around the idea of putting computational thinking and programming into schools. Many have stepped up to answer the important, lingering question: "How?"

The SGD Initiative, a collaboration between the University of Colorado's School of Education, Department of Computer Science, teachers and software company AgentSheets Inc. revolves broadly around making computational thinking accessible (to both teachers and students), and exciting (through game design and STEM simulation creation). The approach solves the *cognitive* challenges of accessibility by going beyond "drag and drop" visual programming. Drag and drop can be compared to spelling and grammar checkers for writing; while useful to help reduce errors, they can't automatically turn anyone into a best-selling author. Visual programming, pioneered in the AgentSheets software, can go far beyond creating programs with the correct syntax. That reduces language-specific frustrations but does not help students understand program meaning, or *semantics.*

Based on over 15 years of research and study, the AgentSheets and AgentCubes environments used in the program have visual programming capabilities, but include extremely powerful debugging tools that help students to create programs that work properly. They have been used with pupils from 7 years old. SGD teachers are supported through training programs, curricular resources and automatic evaluation tools.

Identified as the *Affective* Challenge, SGD acknowledges that students are not interested in programming *per se.* They want to create animations, bring stories to life, and build games. The approach allows students to design and *create* 2D and 3D video games that work in four to five contact hours. Creativity and ownership are key components that make programming exciting. All may create an arcade game, for example, but each pupil can make the game their own by designing and creating unique characters and worlds. AgentCubes extends this potential to 3D games through casual design tools that let users create 3D characters with Inflatable Icons.

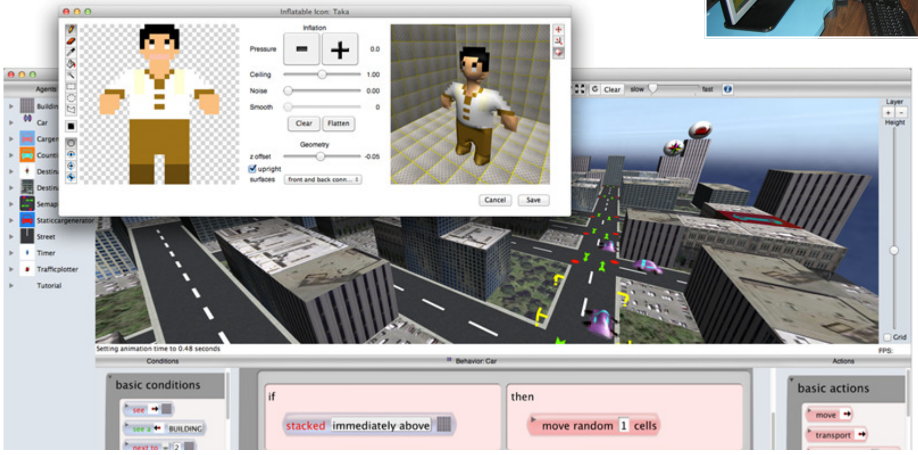
Our results consistently surpass Michael Gove's vision ("we could have 11-year-olds able to write simple 2D computer animations"), even in some of the toughest, poorest and most isolated schools in the US. SGD students can design and build working 2 or 3D video games based on advanced mathematics and artificial intelligence concepts. More importantly, our students, especially girls and underserved minorities, are motivated to pursue further studies in technology and other STEM subjects. More information can be found at http://goo.gl/AYWtYI and http://goo.gl/VuzQgY. Contact Dr Alexander Repenning via alexander.repenning@colorado.edu

SCALABLE GAME DESIGN

With over 10,000 students participating, the SGD Initiative has moved beyond motivated students in after-school clubs and summer camps. SGD partners with school districts to run a one-week unit in classes. Research has evaluated the strategy on four principles:

- *Exposure*. Reaching every student by injecting an easy-to-teach module into computing classes.
- *Motivation*. Carefully balancing challenges through activities that range from simple 'Frogger-like' to advanced 'Sims-like' games.
- *Education*. An approach inspired by latent semantic analysis determines computational thinking and skill transfer between game design and simulation creation.
- *Pedagogy*. Investigating pedagogical approaches and motivation levels across genders and ethnicities. With an optimal pedagogical approach, 35 hours of careful instruction is enough to train teachers to teach SGD curriculum in a gender-friendly way.

In 22 years teaching in diverse Aurora, Colorado, Mark Shouldice tells us he's never seen a more engaging tool. Mark's results are mirrored in our study. Generally 74% of boys, 64% of girls, and 69% minority pupils wish to take a more advanced course at the end of the first.



## LOGICAL THINKING ACTIVITIES TO SUPPORT PROGRAMMING

Students are, typically, very good at learning things. If you write a solution to a programming problem you can walk your students through it and get them to learn it, but the chances are they will be incapable of solving a remarkably similar problem. Programming constructs are like Lego blocks. With a set of prescribed steps you can build a helicopter, truck or police station. But students need to be encouraged to play with the blocks, apply them in different ways and build things of their own devising, without blindly following a recipe.

Before my GCSE students do any programming at all, we spend a number of weeks looking at logical thinking and logical problem solving. Easy examples include the Leapfrog game [http://akidsheart.com/math/mathgames/leapfrog.htm], Towers of Hanoi [http://www.mathsisfun.com/games/towerofhanoi.html] or Bloxorz [http://www.coolmath-games.com/0-bloxorz/]. These interactive puzzles and games encourage logical thinking as well as providing instant feedback to students. As a further challenge, students could be asked to document or describe their algorithms.

There is a wealth of logic puzzles available out there, and there are few better starting points than the Logic Puzzles resource created by Stephen O'Callaghan [http://community.computingatschool.org.uk/resources/82].

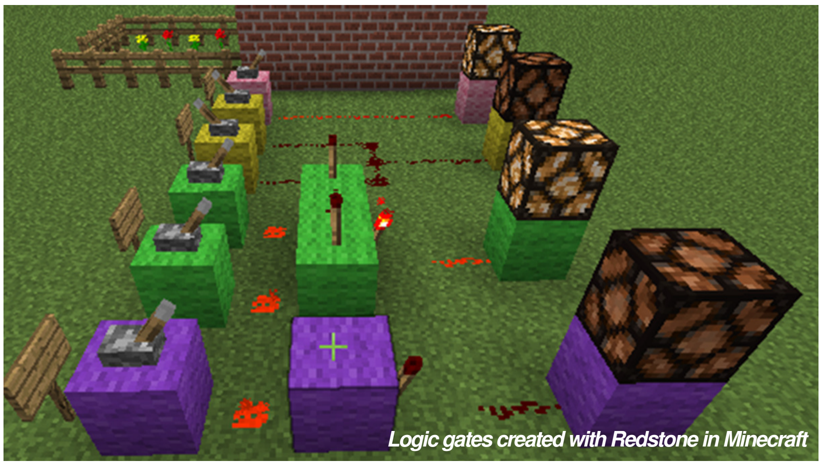
By working through logical challenges that relate to problem solving, decomposition and truth tables, students gain an insight into the strategies involved.

Whilst such activities might not map easily into a scheme of work for programming, the ability to think, to experiment, to identify strategies and to describe algorithms is essential for students who are looking to become better programmers. And, perhaps more importantly, those who become better programmers will be better equipped to think, to experiment, to identify strategies and to describe successful algorithms in whatever field they find themselves in the future. *Mark Clarkson*

## THE POWER OF REDSTONE USING MINECRAFT IN CLASS

**Minecraft is a phenomenon. Over 20 million copies are registered, not including the free version available by typing "try Minecraft for free" into Google. But it's just a game, right? Not really, says Ben Smith of AKS in Lytham St Annes.**

There are two classic modes of use in Minecraft - survival and creative. This article focusses on creative. Think of a giant virtual box of Lego. It has the most potential for teamwork and education. It removes the need to fight off beasties which plague the survival mode and allow pupils full control over the powerful command line interface. I've been teaching Minecraft to KS3 for half a term now. I normally split the lesson into two interlaced activities. I introduce each logic gate with a quick explanation then ask them to recreate it in their own world. Once complete I ask them to build me their dream house with a swimming pool, conservatory & rooftop terrace. The real challenge is to get the logic gates to control the features of their house to make a fully automated home.


*Logic gates created with Redstone in Minecraft*

The logic gates in Minecraft have become a sub culture in the genre. A YouTube search will reveal a gluttony of podcasts explaining how to make all manner of gizmos using sticky pistons to push and pull objects based on a virtual current supplied by the key ingredient - Redstone. The real draw for me is the ability to model logic circuits, binary maths and even ultimately create a fully functional low level computer which the pupils can walk round within the 3D world. Basic AND, OR and NOT gates lead quickly to half adders and full binary calculators. If... Then... Else is a short step away. I've been playing a month and even the free online version allows anyone to quickly & easily model a logic driven binary addition. The raspberry Pi version shows even more potential with the ability to create sequences of instructions and loops in basic python - I was shocked at how simple it was to create a 3D traffic light sequence.

I have joked with colleagues trialling the program about "Griefing" - a term I wouldn't have even known how to pronounce a month ago but which has become as ubiquitous to my vocab with the kids as "cool" or "sick". I used to think Minecraft was just another silly game that I would steer clear of - much like my aversion to Candy Crush or Farmville. I now see it as a vehicle for introducing algorithms to pupils in a 3D environment in much the same way as Scratch does in 2D. You really should try it. It'll cost you nothing to put your kids on for a lesson and within 10mins they'll be desperate to show you what they can achieve.

# GETTING A HANDLE ON A RASPBERRY PI

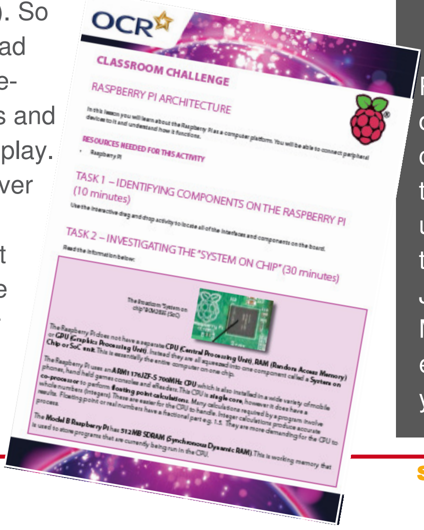
**Clive Beale, Director for Educational Development at the Raspberry Pi Foundation introduces some excellent free classroom resources produced by OCR.**

The Raspberry Pi is a weird little beast designed to encourage play, experimentation and physical computing. But making a bare-board, system-on-chip, Linux based computer with SD card storage has had its problems. People who have only ever been exposed to mass-consumer hardware and software often take some convincing that the Raspberry Pi is a proper computer and not just some esoteric, specialist circuit board. Times are hopefully changing but in the meantime we want to help people break away from the sealed-box paradigm and help teach computing in creative ways. One way we are doing this is by producing resources.

"Resources" sound pretty mundane to the average man on the Bridgend omnibus but to teachers resources are currency: buy, sell, swap, donate, "borrow". If teachers were boxers, all a trainer would have to if they were KO'd would be to whisper, "Free resources!" and they'd leap up shouting "Where, where?!" and "Are they Creative Commons?!" It's hard-wired into their cerebellum. It's not that teachers aren't capable of making their own resources but there's the "T" word. Yes—"terrapin". (Or perhaps "time", I can't read my notes.)

Quality resources can be the difference between teaching something new this term or sticking with what you've got. And in computing, the "new" has to be the way to go every time. So one of our main priorities is to help teachers who want to use Raspberry Pis in their classroom. We've also been working with various partners. For example, we've been working with OCR on all things related to teaching computing, most recently the Cambridge GCSE Computing Online. Check out http://goo.gl/NIs0e3. OCR are also producing a set of resources for the Pi to liven up your lessons. The first of these can be found at http://goo.gl/jMCQWm. Currently there are four self-contained "recipe cards" that show you how to use the general purpose input output pins (GPIO) to connect to the real world. You can make jelly babies sing, flash LEDs and control the Pi via Twitter.

There are also five Classroom Challenges that have detailed lesson plans, student worksheets and even drag and drop exercises. They highlight how powerful the Raspberry Pi can be in teaching computing concepts, from hardware architecture to networking. Every training day drags out the old Confucian chestnut, "I hear and I forget. I see and I remember. I do and I understand" for a reason. Actually overclocking a machine and then benchmarking it will stick with the student. Good luck doing that on the school PCs! I actually teach this lesson using a cheap infrared thermometer to measure the temperature of the processor. It gets their attention every time and is a jump off point for discussions on everything from computer architecture to physics to computer gaming (why does your laptop get noisier when you're playing a game?). So download some resources and have a play. You never know where it will take you. Or your students.



OCR

CLASSROOM CHALLENGE

RASPBERRY PI ARCHITECTURE

In this lesson you will learn about the Raspberry Pi as a computer platform. You will be able to connect peripheral devices to it and understand how it functions.

RESOURCES NEEDED FOR THIS ACTIVITY

- Raspberry Pi

TASK 1 – IDENTIFYING COMPONENTS ON THE RASPBERRY PI (10 minutes)

Use the interactive drag and drop activity to locate all of the interfaces and components on the board.

TASK 2 – INVESTIGATING THE "SYSTEM ON CHIP" (30 minutes)

Read the information below:

The Broadcom "System on chip" BCM2835 (SoC)

The Raspberry Pi does not have a separate **CPU (Central Processing Unit)**, **RAM (Random Access Memory)** or **GPU (Graphics Processing Unit)**. Instead they are all squeezed into one component called a **System on Chip or SoC unit**. This is essentially the entire computer on one chip.

The Raspberry Pi uses an **ARM1176JZF-S 700MHz CPU** which is also installed in a wide variety of mobile phones, hand held games consoles and eReaders. This CPU is **single core**, however it does have a **co-processor** to perform **floating point calculations**. Many calculations required by a program involve whole numbers (integers). These are easier for the CPU to handle. Integer calculations produce accurate results. Floating point or real numbers have a fractional part e.g. 1.5. They are more demanding for the CPU to process.

The **Model B Raspberry Pi** has **512MB SDRAM (Synchronous Dynamic RAM)**. This is working memory that is used to store programs that are currently being run in the CPU.

## WHAT EXACTLY IS A RASPBERRY PI?

Computer Science pioneer Seymour Papert wrote, *"The role of the teacher is to create the conditions for invention rather than provide ready-made knowledge"*. The Raspberry Pi, costing about the same price as a textbook (£18) is one such tool. It encourages children to tinker with little financial risk, offering massive educational value in return. The unconventional, bare-bones appearance of the Raspberry Pi computer frequently prompts more questions than it answers. In education, that is surely a good thing.

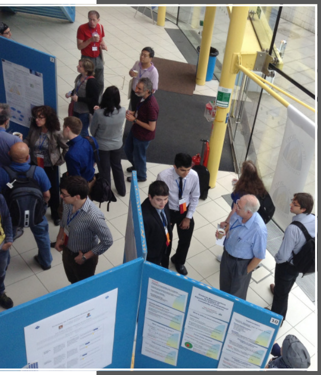
Much like baking a cake, there are an amazing wealth of recipes online and different ingredients you can purchase if you want to add to those already in your 'larder'. Making it work is the next challenge. You will need an operating system, eg. Raspbian, on an SD memory card. You can either go for the home-baked approach or save a little time by buying a pre-loaded SD card.

I suggest that teachers launch a classroom discussion by first asking your class to *"Agree a definition of what a computer is"*. Many will say it must have a keyboard and mouse. What about tablet devices? The next discussion could be *"Are computers taking over the world?"* as it is dominated more by embedded computers tracking conversations, spending, viewing habits and more.

I regularly meet teachers at the Raspberry Jam events I attend, driven there in the main by their curiosity. 300 teachers attended this years Raspberry Jamboree to understand the educational potential of the Pi computer. Next year's Jamboree, planned for Feb 28 – Mar 2 2014 looks certain to be even bigger, perhaps we will see you there? *Alan O'Donohoe*

## ITICSE CONFERENCE INCLUDES DEDICATED TEACHER DAY

The 18th Annual Conference on Innovation and Technology in Computer Science Education was held at the University of Kent in July. This conference attracts academics from around the world and this year, for the second year running, included a special "teachers' day" with a focus on Computer Science in school. Our own Simon Peyton Jones gave the first keynote which generated a lot of interest from other countries; this was followed by a panel discussion looking at the progress made in USA, Australia, Israel and UK on transforming Computer Science in school. The issues we have in the UK are replicated throughout the world as other countries examine how best to introduce computational thinking to children in school.



There were several posters and papers focusing on issues relating to school including assessment of KS3 programming skills, gender and inclusivity issues, game programming, and teachers' perspectives on what motivates students. Some of the work carried out in universities on particular ways of supporting beginning programmers at undergraduate level is also very applicable to students learning to program in school, so hearing about developments here was extremely interesting.

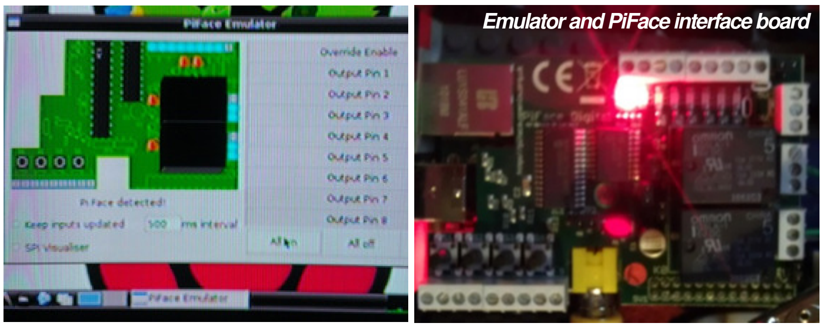
A highlight for me was the keynote on the Bloodhound project which aims to interest students through the development of a car attempting the land speed record. The conference was a great opportunity to meet others working in school Computer Science and I gathered many (too many!) new ideas for future projects. Attending the conference reinforced my view that it would be good to develop more research amongst the CAS community and I hope that a model for achieving this can be established in the near future. Do contact me sue.sentance@computingatschool.org.uk if you are a teacher interested in carrying out an action research project in your school. *Sue Sentance*

## GETTING PHYSICAL WITH PIFACE PROJECTS

**The Raspberry Pi is very versatile. CAS Master Teacher, Chris Swan, outlines ways to introduce physical computing and control using the GPIO pins.**

By using the GPIO (General Purpose Input Output) pins on a Raspberry Pi, simple components such as LEDs and buzzers can be controlled by writing programs using the Python GPIO libraries. The OCR "recipe card" ideas mentioned on p21 provide a very interesting set of activities. Components are cheap: LEDs, resistors and jumper wires are a few pence each and the "singing Jelly Baby" activity is a hoot! Students learn about simple components and circuits. Once the basics have been mastered, they can switch on their inventing genes. I recommend perusing the Pi [http://www.raspberrypi.org] and Adafruit [http://goo.gl/hZA9Bs] websites and Carrie Philbin's Geek Gurl Diaries blog [http://goo.gl/ZsM1MJ] for more fantastic ideas. Invest in breadboards for prototyping as these allow temporary circuits to be constructed.

So, what do you do if you are planning on building a Pi controlled car with flashing lights, tweeters, sub woofers and a Jelly Baby driver? You'll need an interface board. My personal choice would be the Piface Digital developed at the University of Manchester School of Computing by Dr Andrew Robinson and his team: http://pi.cs.man.ac.uk/interface.htm. It is robust and simple to set up, connect devices to and program. The "Pifaced Raspbian" operating system contains: an emulator for testing ideas and allows direct control; the PFIO libraries for Python and a special version of Scratch. My Year 9 students very quickly wrote test programs using Scratch to prototype ideas without getting bogged down in Python syntax. The Piface has LED indicators on the 8 output connections and also has two relays which permit safe connection to devices such as motors without risking damage to your Pi. I used old Lego motors driven by batteries in a Lego chassis to make a simple buggy.



*Emulator and PiFace interface board*

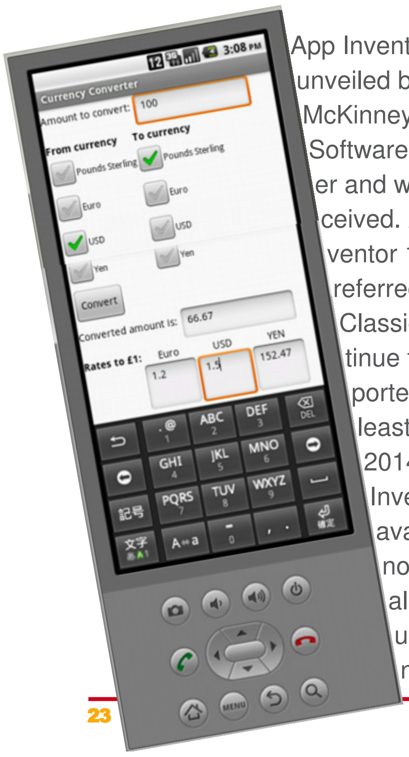
For input, switches are the simplest option. Piface has 4 onboard input switches but you can connect your own external ones too. You will need to add an Analogue to Digital Convertor (ADC) if you wish to connect analogue sensors. I love the "hackability" of the Pi. Developing a project taps into creativity, ingenuity, resilience, problem solving, planning and deep thought. Everything is customisable. I have allowed my students to "play" and try ideas out. So far, nothing has broken: not an SD card corrupted or circuit shorted. Students take real care over the kit. I can also see how the Raspberry Pi could easily go "cross curricular" and be used in other lessons. For such a little box, it's having a huge impact on my teaching.

# ANDROID APPS FOR BEGINNING CODERS

**Uptake of App Inventor is growing fast in the UK. Trevor Bragg, from Southfields Academy in South West London reports from the recent MIT App Inventor Summit.**

App Inventor's user base has grown 10-fold since April 2012 when it transferred from Google to MIT. Hal Abelson declared the Summit as the 'Birth of Inventing Personal Mobile Computing' and set the agenda for how to teach mobile computational thinking. Hal was giving the opening welcome and was on hand throughout listening for new ideas and answering questions. Hal was the founder of App Inventor whilst on sabbatical at Google and continues to lead App Inventor development. It is being used in the US in extra-curricular clubs, High Schools and Universities.

Professor David Wolber has produced some excellent resources including an online text book and a course that he has run with undergraduates of non-Computer Science majors. You'll find them, and lots of other useful free resources at http://www.appinventor.org/. Another useful source of tutorials can be found at http://appinventor.mit.edu/explore/tutorials.html



App Inventor 2 was unveiled by Andrew McKinney, its Lead Software Developer and well received. App Inventor 1, now referred to as Classic will continue to be supported until at least July 2014. App Inventor 2 is available now as an alpha product for pioneers and will be released in the Autumn. Go on, try it now at www.ai2.appinventor.mit.edu It's not just an upgrade, but a new product with the same screen design. A new blocks editor works in the web, not requiring Java, and it has an emulator like the current version. Those at the CAS Teacher Conference have already seen the Blocks editor, called Blockly, demonstrated by Neil Fraser from Google. It promises many improved features. The Blocks Editor loads quickly from screen design without needing Java. The Designer and the Blocks Editor both run in the browser and are linked. It's easier to setup as Java is not required, so saving time on school networks. The connection to the emulator is quicker. Local Variables are available for procedures. The colour of the blocks is much clearer than the pastel colours of classic App Inventor. Available blocks are less cluttered as properties are hidden in pull-downs and fewer If blocks are initially presented, though all are available using mutators

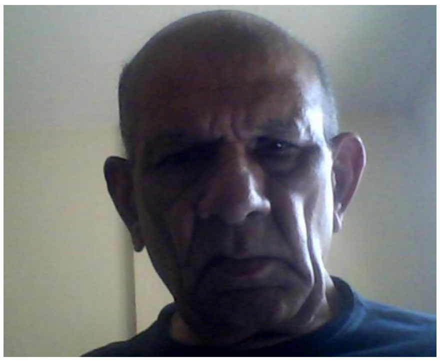
There are some current limitations. Classic App Inventor is not compatible with App Inventor 2, although MIT plan to offer upgrading apps soon. There is little documentation as yet, nor any zoom capability within the blocks editor, like the one in Classic. However, I am excited by App Inventor 2 and will trial it in the classroom from September, in parallel with classes using Classic. The new product is continually going through changes and therefore I recommend users not wishing to pioneer stay with App Inventor 1 for the moment. It is stable and has excellent educational resources readily available. For further information you can contact me via trevor.bragg@computingatschool.org.uk

## ARE YOU READY FOR THE NEXT CIPHER CHALLENGE?

The National Cipher Challenge run by Southampton University costs nothing and runs annually, from September through to December/January. Harry, the organiser and moderator, makes it really easy to sign up. Each week's challenge is issued on a Thursday at 3.30pm. As the weeks go on the challenges get harder, that's all part of the fun. I've been taking part in the cipher challenge for 5 years now and have loved every part of every challenge, regardless of whether or not I can do them. For those few months every Thursday I would rush home to get an early start on that weeks challenge, always beginning by searching for patterns and, in the later stages failing to find any! Every time I finished part A I got a sense of relief - only to remember the harder more important part B needs to be cracked as well, all before midnight (for maximum points)! The challenges are woven into a story which spurs you on. Though I have not, as yet, solved the final challenge, I always check to see the end of the story.

I have had to learn about Ceasar shift, beaufort, and vignere ciphers, transposition, substitution and frequency analysis. Last year, I progressed through challenges 1 to 8, adopting various techniques that I had not come across before, without being aware of the different methods I'd learnt until I looked back at the end. I would whole-heartedly recommend the challenge to anyone, regardless of ability. It is really fun and the prospect of prizes help! I don't think I can do the competition justice as it really is a unique and rewarding undertaking. *Michael Harkness*

Michael is just starting Year 12 at Queen Elizabeth School, Cumbria. More details from http://www.cipher.maths.soton.ac.uk/



# FROM BITS TO CHIPS: CONCEPTS BEHIND A DIGITAL COMPUTER

**Having a conceptual understanding of all aspects of digital systems is important. Building that strong foundation begins at school. Arun Warhadpande, a retired radar data processing specialist offers a holistic introduction for teachers new to computing.**

The most popular computer system these days is the **Stored Program Digital Electronic General Purpose Computer.** These machines have been made extremely user friendly. Almost all of us are familiar with some aspects of this system. I am making a humble attempt at giving a holistic picture of the system. Each aspect though is a discipline in its own right.

## COUNTING AND PLACE VALUE

People the world over have been devising different methods of counting since very early days. It was only after the use of **place value system** and the concept of Zero (0) supposedly used by *Aryabhatta* in the fifth century, that the current system evolved. In this system the base could be considered to be any selected number of digits. For example in **Decimal** the maximum digits (Base) is 10 (0 to 9). Adding 1 to 9 we get 10. Place values in Decimal are:

| | | | | | |
|---|---|---|---|---|---|
| **Base and exponent** | $10^4$ | $10^3$ | $10^2$ | $10^1$ | $10^0$ |
| **Place value** | 10000 | 1000 | 100 | 10 | 1 |

In **Binary** the maximum digits (Base) is 2 (0 to 1). Adding 1 to 1 we get 10. Binary place values are:

| | | | | | |
|---|---|---|---|---|---|
| **Base and exponent** | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| **Place value** | 16 | 8 | 4 | 2 | 1 |

So in the place value system one can choose any base. 2,8,10,16 are the popular bases for binary, octal, decimal, hexadecimal respectively. Children might want to consider why an hour has 60 minutes and a minute 60 seconds? Why is a circle divides into 360 degrees, a dozen consist of 12 items or why the decimal system is most popular the world over?

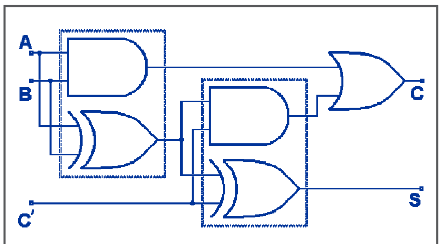
## DIGITAL COMPUTERS

In these machines all inputs are sensed by a **sensor**, converted into an electrical form by a **transducer** and then **digitized** by an analogue to digital converter. Digitization involves breaking down the input into steps and keeping a count of the steps. These are then stored and processed. The digital output so obtained is then given to an output system. The output system does the reverse process of an input system i.e., a transducer converts the digital input into analogue electrical form which activates the output device.

## THE RELEVANCE OF BINARY AND BOOLEAN ALGEBRA

It may be noted that in digital computers, the count is kept with the help of switches and processing done by controlling switching sequence. Switches have only two states, ON and OFF thus implying that by using a binary system we can achieve anything. Binary cannot be more difficult than decimal as counting to one is much easier than counting up to nine.

**Boolean Algebra** is an algebra in which an element can have only one of two values, you may term them, TRUE and FALSE, 0 and 1, OFF and ON etc. The operators are only three, namely OR, AND and NOT. Logic gates are designed to perform these operations. It is amazing to observe that **any logical argument can be represented and arithmetic operations performed by suitably interconnecting these gates.**



*A 'full adder' adds 2 bits and a carry from the previous column. Built from OR, XOR and AND.*

We can develop arguments by representing statements using Boolean variables and interconnecting them with Boolean operators AND, OR and NOT. All logical arguments and arithmetic operations can be performed by Boolean expressions. By suitably combining the basic OR, AND and NOT gates we can get the EXCLUSIVE OR , NOR and NAND gates. Playing around with logic gates for implementing Boolean expressions is quite fascinating.

With the current developments in semiconductors it is possible to interconnect millions of logic gates on a single chip. A single chip is thus capa-

## LOGICAL ARGUMENTS AND ARITHMETIC

- This class is empty. This statement is true if teacher is not there is true AND students are not there is true. This class is not empty is true if teacher is there OR students.
- By a combination of gates we can make circuits to add bits (Adders)
- Subtraction is done by negative numbers using a representation called 2's complement for the negative number. This can then be added it to the number to be subtracted from.
- Subtraction can therefore be done by circuits for addition. Since multiplication is repeated addition and division is repeated subtraction. Circuits for addition can perform all arithmetic operations.

## LOW LEVEL AND MACHINE DEPENDENT

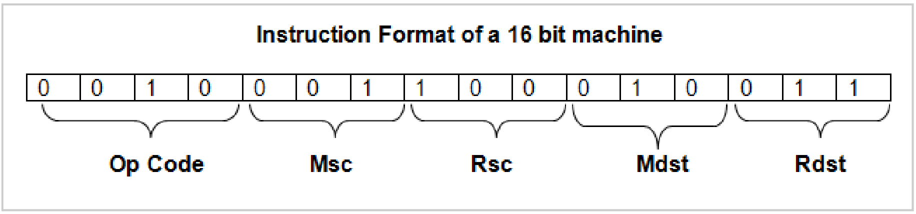
- Machine languages are different for different CPU designs. These are also called the lowest level languages because they are closest to the machine.
- Instructions in machine language can be directly executed by the machine.
- The binary code if grouped in threes (octal) or fours (hexadecimal) it is **easier for humans to remember.** No machine actually follows octal or hex, it is only easier than binary for humans to remember.
- While using machine code, the programmer has to manage the resources of the computer as well.

ble of performing logic and arithmetic functions. This chip could form the Central Processing Unit of a computer system. Millions of logic gates integrated in a single chip for processing is a **microprocessor**.

## MACHINE LANGUAGE

Every design of CPU will have its unique **Instruction Format** for performing operations. Instruction Formats will have some bits for the **operation code (Op Code).** Each operation (add, subtract, compare etc) will have its unique combination of bits. Other bits in the format are reserved for **address specifications.** Address specifications will enable fetching of the operands (data) on which the operation is to be performed.

Let us for example consider a model 16 bit machine. The instruction format may look like the example below.



We could have the most significant four bits as the Op Code, thus providing 16 unique combinations for operations like add, subtract, compare etc. The next three bits are for addressing modes of the source operand (MSc) (8 combinations) and the next three for specifying the Register of the source operand (Rsc) (8 combination). The next six bits are for destination operands.

A machine language program would contain several such instructions. If you know the machine language of a processor, you can write a program in binary store it in the memory and execute it by accessing the memory. No systems program is required.

Using mnemonics for the operations to be performed and describing the address specifications would ease the programmer's effort. This is what is done in **assembly language.** It is also machine dependent and will require the development of an **assembler** to convert the assembly language instruction to machine language.

## MAKING THE MACHINE USER FRIENDLY

We are always working to make life as easy as possible. The example machine will need a lot of specialization for us to use it. Besides binary (not very difficult) one needs to know the Instruction Format (not difficult either). The addressing modes and utilization of registers are a little tricky and keeping a track of resources is rather troublesome. A machine like this is unlikely to become popular and people would rather find alternate methods for solving their problems.

In order to make the machine user friendly a host of different software is developed. Some of it is mentioned below:

- Operating Systems provide a variety of functions to manage the resources such as the peripherals, memory, storage and processor time.
- Language convertors (see sidebar).
- Diagnostics programs
- Utilities that will carry out a variety of 'housekeeping' tasks.

This category of software does not solve user defined problems but serves to make the machine user friendly. Collectively it is referred to as Systems Software.

In order to make the machine even more user friendly many different common user software packages have been developed like the word processors, spreadsheets, presentation packages, paint etc. The are also numerous packages to help professionals in Architecture, Law, Music and so on.

## THE FINAL SYSTEM

We have just glanced through the various stages that are involved in providing us with a system which we now commonly refer to as a computer. It may be appreciated that the rapid growth in semiconductor technology is to a great extent responsible for bringing these machines from research institutions to the lap of the common man and woman.

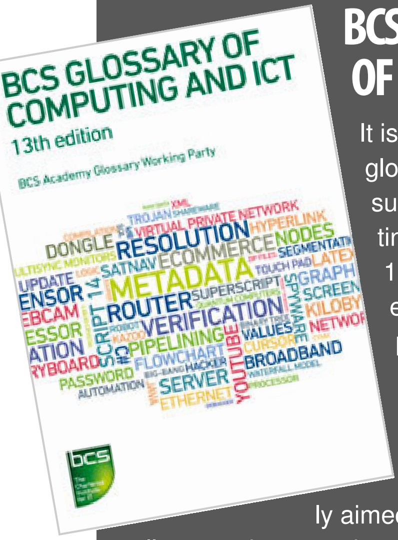
## HIGH LEVEL LANGUAGES AND CODING

Over the years a large number of machine independent languages have been developed. These languages are a lot closer to the commonly understood languages and are machine independent. To write programs in these languages the programmer need not know the architecture/organization of the machine. The development of these high level languages meant programming could be oriented on the user rather than the machine

Programming is the term applied to developing programs to solve problems. The language used could be anything from the machine language to any high level language. Coding involves writing instructions in the syntax of the language chosen for programming.

The syntax of a high level language is machine independent but the language convertor ( a **compiler or interpreter**) for the language is machine dependent.

## BCS GLOSSARY OF COMPUTING


BCS GLOSSARY OF COMPUTING AND ICT
13th edition
BCS Academy Glossary Working Party
bcs

It is difficult to get a glossary right in a subject like computing (or ICT!). In its 13th edition however this glossary promises to catch up with the ever-changing and expanding terminology. Primarily aimed at school and college students, and useful to those early on in higher education, this is a compact and neat dictionary of terms compiled in a single source.

The glossary is divided in four main sections: how computers are used, what they are made up of, how they communicate and how they work (architecture). Each of the sections are further divided up into subtopics that are relevant to the field. This structure serves an additional purpose of trying to build up knowledge in steps - taking one concept and developing it further one by one. For example, the section on 'sound processing' starts from the basics with frequency modification and takes it all the way to sound formats, streaming audio and mixing and sequencing. Each concept is explained as a succinct description using accessible basic terms.

The book is appended with list of acronyms, file format definitions and binary logic descriptions. This is very welcome.

I recommend this book to those who are new to the subject. Those more familiar with the subject may find that some of the terminology has moved on and perhaps the book is too grounded. However, to those who are new, and from a pedagogic perspective, this is a good start and a handy book to have on the shelf.

BCS Glossary of Computing and ICT (13th Edition), BCS Academy Glossary Working Party, BCS, 470pp, ISBN 9781780171500 £19.99 Reviewed by Dr Siraj A. Shaikh, Senior Lecturer, Coventry University. Further information: www.bcs.org/books/glossary

## LOOKING BACK: END OF YEAR REPORT FOR GCSE COMPUTING

**At the end of the first year delivering OCR's GCSE Computing course to years 9,10 and 11 at Stourport High School and Sixth Form Centre, Worcestershire, Christine Swan asked her students how they had found the course.**

Every response, without exception, said that they had enjoyed the experience. Many admitted that they had found it challenging but interesting. Students undertook one controlled assessment task using the Little Man Computer assembly language programming simulator. I had thought that they might grow to enjoy this but a few students commented that they had found this tricky to master. The second controlled assessment that we did was a combination of Scratch and Python programming.

Students particularly enjoyed creating simple games on their journey through learning Python. We started with simple number guessing games and progressed to battleships, tic tac toe and simple graphics and sound using Pygame. Considering that students had only experienced Scratch when they joined the course, they learned Python very quickly and by the end of the year were proficient at spotting errors and writing their own programs from a blank canvas. I felt very proud when I looked around the room and saw groups of students working together on their ideas at the end of the year, just having fun coding.

One student commented that studying GCSE Computing had a wider impact: " I found the course very interesting as it was a new experience as I had never done anything like this before. At first, I found it quite hard but as I learned more and my experience grew I felt comfortable tackling anything that was thrown at me – within reason! I think that what I have learned through studying Computing will help me to think more logically and approach problems in a different way."

Another said: "Studying GCSE Computing has given me the skills to begin coding my own games. I have learned Python which I know is widely used in the software industry and has allowed me to do something that I love doing – programming!" One of the simplest responses I received was: "Computing is the best subject in the world!"

Reflecting back on the year it certainly does feel like we have managed to pack a huge amount of learning into it. Coding will still be offered as an after school club to allow them to continue to practice their and refine their skills. I'm going to use my ex-GCSE students as mentors to those with less experience. I've already checked my group lists for next year. It looks like another busy year ahead!

## ADA LOVELACE DAY: A CHANCE TO CELEBRATE

Ada Lovelace Day will be celebrated on October 15th. It provides a good opportunity for teachers to bring the achievements of Ada, and her unique contribution to the development of Computing. It's also a chance to highlight and celebrate the impact of the many other women in Science, Technology, Engineering and Maths. You can find out more information about events already being planned by visiting http://findingada.com/

## ADA AT THE EDGE: PROMOTING COMPUTER SCIENCE FOR GIRLS

**Over sixty year 8 girls from Maricourt, St Bedes and Maghull High Schools attended a free activity packed day at Edge Hill University designed to enjoy computing in a mostly female, definitely fun atmosphere. Dawn Hewitson reports.**

Held last June, the program appealed to girls' design aspirations by focusing on creativity through technology and computing. The sessions were run by University lecturers ably assisted by IT and Computing trainee teachers and NQTs. Funding for the event was provided by CAS Include.

There were a host of activities and hands on workshops including Exploring Stereotypes, Looking at Wearable Technology (delivered by Colette Giblin), Designing Inspirational Apps Using App Inventor (delivered by Paula Beer) and Creating Small Computer Games in Stencyl which I delivered.

The day went very smoothly and feedback throughout has been extremely positive. As well as structured questions in the evaluation, the participants were encouraged to suggest how the day could be improved and asked why they would recommend the day to another school. The most frequently occurring answer to these questions was that the girls wanted more…… longer sessions, more sessions, even more activities.

Several of the girls made the point that they could achieve even more if there had been enough time to arrange longer sessions and workshops.

Overall the feedback showed it was fun, interesting and educational. Comments from the girls included "It shows that some jobs aren't just for men", "It is inspiring and I don't think people realize that it's not that hard" and "It was fun and interesting to hear about women in technology".

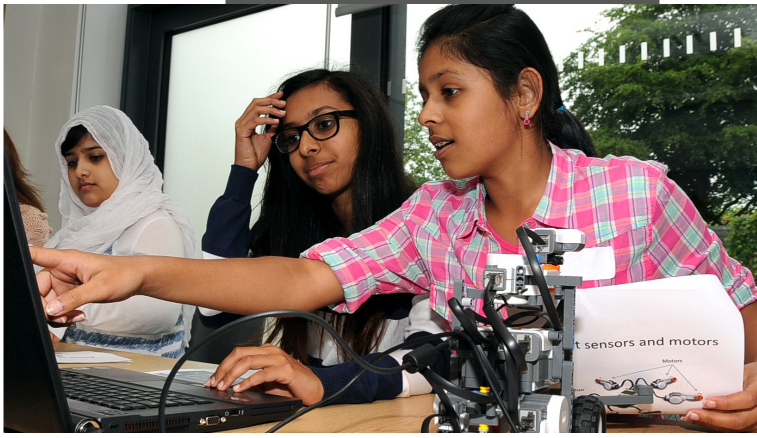
Radio Lancashire asked for an interview so Colette Giblin and I went to the Blackburn studio to describe the girls' reaction to the day and explain why the issue of underrepresentation of girls in computing and technology is so important.

More than a hundred students aged 11 to 13 years enjoyed #define last June - a free event run by #include at Rugby School. It aimed to encourage girls and students from minorities to take up Computer Science as a subject. The day began with a talk from multi-award winning designer Laura Kalbag. Students enjoyed workshops in the school's IT rooms and the new Modern Languages labs. Activities included making games in Greenfoot, programming Python and lighting LED lights using the Raspberry Pi.

The day was a huge success. Students created mobile apps and programmed Lego robots to move around a classroom, transformed into a robot garden. Some made controllers for their software from everyday objects including the floor, themselves and even Blu Tack in the Makey Makey workshop. A teacher from The Kingswinford School in Dudley commented: "The event was jam packed with things to do. It was exceptionally well run with a timetable of events that allowed for the broadest experience possible for all. A great day for the students and accompanying staff." Generous sponsorship from OCR and Pimoroni enabled winning students from each workshop to take away a Raspberry Pi to practise what they had learned during the day. *Laura Dixon*

## PROGRAMMING CHALLENGE 4 GIRLS

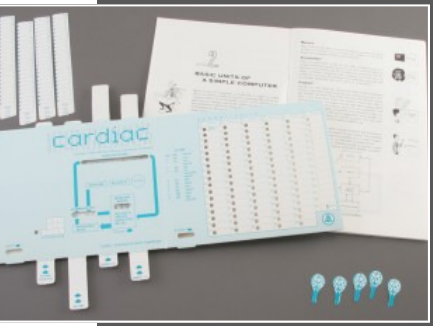
At Fakenham High School all Year 9 girls did programming for a day. We played around with Alice. It was very funny, as many people in my group had heads, legs, arms and bodies flying off the screen everywhere. Task 1 was to design a marine biologist taking a picture of a fish, getting bitten by a bigger fish and swimming off screen. Task 2 was to create a skate girl skateboarding up ramps and back down again. Both were harder than they sounded but the whole day was very interesting and funny. A huge thank you to Mrs Gray for organising it. *Laura (Y9 student at FHS)*



Hans Pufal is a software engineer and computer historian. Like many Hans caught the computer bug when in his teens and taught himself to program. Unlike many that was in the mid 1960's when computers filled an entire room. His interest in computer history leads him to write emulators so that he can run and study old software. The development of VISIAC is just a step removed from that passion.

## THE EARLY HISTORY OF COMPUTER SIMULATORS

In 1968, Bell Telephone Laboratories published CARDIAC, a CARDboard Illustrative Aid to Computation designed to teach schoolchildren how computers worked. The kit consisted of a die-cut cardboard 'computer' and instruction manual. Once assembled you could write programs and execute them by sliding cards.



It had 100 memory locations, worked in base 10 and used a set of 10 instructions, allowing it to add, subtract, test, shift, input, output and jump. The instruction manual covers topics such as Basic Units of a Simple Computer, Flow Charts, Loops, Instructions, Data, Addresses, Multiplication, Shifting Digits, Bootstraps, Subroutines, Assemblers and Compilers.

It was a wonderful tool and my A Level students still enjoy using a replica. Manually manipulating the sliders, doing the arithmetic and writing values in locations bring home to students the low level steps required to accomplish even simple operations. You can watch one being operated at http://goo.gl/MvW8YH, find plans to build your own at http://goo.gl/JwKJk6 and view an original in the Bell Labs archive at http://goo.gl/Det1Jv. Online simulations of Cardiac also exist, and you'll find links and further information on Wikipedia. *Roger Davies*

# VISIAC: A TOOL FOR EXPLORING COMPUTER SCIENCE CONCEPTS

**The VISIble Academic Computer (VISIAC) is a web based emulation of a simple computer designed to be used as a computer science teaching aid. Developer Hans Pufal outlines its specification and remarkable capabilities.**

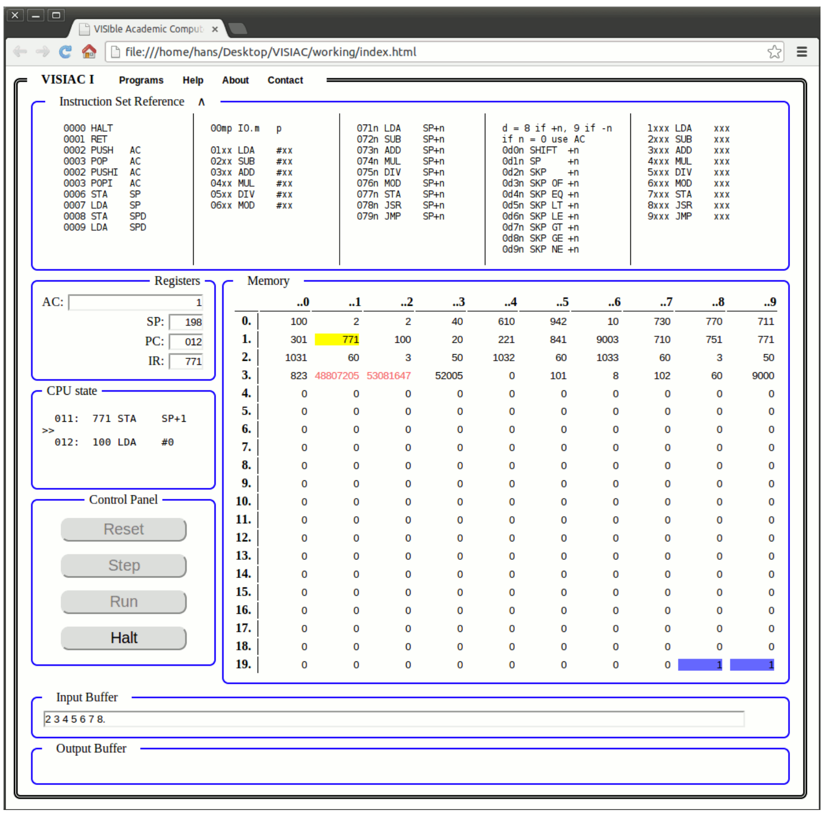
All programming languages are abstractions of the processor on which their programs ultimately run. For that reason, an understanding of a processor and its operations can be helpful in mastering any computer language. Algorithms are also abstractions, the visualisation of their execution can be a valuable aid in understanding them. VISIAC uses modern web based technology to significantly update earlier teaching aids such as the Bell Labs CARDIAC and Little Man Computer (see left). Unlike these predecessors, VISIAC is a fully capable computer designed to be used in the teaching and study of many facets of computer science. The implementation as a JavaScript web application makes it easily accessible to anyone with a modern web browser regardless of host machine architecture or operating system. It has been demonstrated on the Raspberry Pi, Apple iPad and on Android tablets as well as on the more traditional Windows and Linux PC's. VISIAC can be started by browsing to its online web page, or the entire application can be downloaded and run offline on the host system. Currently being tested, it will be available soon from http://pufal.net/VISIAC/

## THE USERS VIEW

Multi-panel, on-screen help detailing all aspects of the VISIAC is provided. The help panel can be opened in a separate window to allow consultation while operating VISIAC.

Register and memory are displayed in their respective panels and the contents may be freely edited. A set of control buttons allow the user to start execution of a program or to single step instructions. When running, the VISIAC continuously updates the register and memory content displays. As each instruction is executed, changes in register contents and memory can be observed. By continuously highlighting the memory location of the currently executing instruction, the concept of flow of control takes on an almost physical reality. The stack is also highlighted and its use during the execution of an algorithm can be clearly seen. All numbers are displayed in decimal for easy interpretation.

The first simple programming exercises will entail typing numbers into memory. Educational as this intimate interaction with the raw processor may be it soon becomes tedious and error prone. More elaborate programs can be developed using the built in symbolic assembler which provides the first level of abstraction of the processor. Access to the assembler is provided by the toolbar item labelled programs. Here a set of pre-written examples are provided which the user can use as models to develop their own programs. Editing can be done from within VISIAC or by using an external editor and the program source copy/pasted into VISIAC. The single pass assembler provides for symbolic addressing of memory, forward reference resolution and operands specification as simple values or complex expressions. The assembler is invoked by pressing the load button, any errors in the program will be reported and the load aborted. Continuity with numeric programming is provided by



Instruction Set Reference ∧

```
0000 HALT          00mp IO.m   p       071n LDA    SP+n     d = 8 if +n, 9 if -n    1xxx LDA    xxx
0001 RET                               072n SUB    SP+n     if n = 0 use AC         2xxx SUB    xxx
0002 PUSH    AC    01xx LDA    #xx     073n ADD    SP+n     0d0n SHIFT  +n          3xxx ADD    xxx
0003 POP     AC    02xx SUB    #xx     074n MUL    SP+n     0d1n SP     +n          4xxx MUL    xxx
0002 PUSHI   AC    03xx ADD    #xx     075n DIV    SP+n     0d2n SKP    +n          5xxx DIV    xxx
0003 POPI    AC    04xx MUL    #xx     076n MOD    SP+n     0d3n SKP OF +n          6xxx MOD    xxx
0006 STA     SP    05xx DIV    #xx     077n STA    SP+n     0d4n SKP EQ +n          7xxx STA    xxx
0007 LDA     SP    06xx MOD    #xx     078n JSR    SP+n     0d5n SKP LT +n          8xxx JSR    xxx
0008 STA     SPD                       079n JMP    SP+n     0d6n SKP LE +n          9xxx JMP    xxx
0009 LDA     SPD                                            0d7n SKP GT +n
                                                            0d8n SKP GE +n
                                                            0d9n SKP NE +n
```

Registers

| | |
|---|---|
| AC: | 1 |
| SP: | 198 |
| PC: | 012 |
| IR: | 771 |

CPU state

```
  011:  771 STA    SP+1
>>
  012:  100 LDA    #0
```

Memory

| | ..0 | ..1 | ..2 | ..3 | ..4 | ..5 | ..6 | ..7 | ..8 | ..9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0.** | 100 | 2 | 2 | 40 | 610 | 942 | 10 | 730 | 770 | 711 |
| **1.** | 301 | 771 | 100 | 20 | 221 | 841 | 9003 | 710 | 751 | 771 |
| **2.** | 1031 | 60 | 3 | 50 | 1032 | 60 | 1033 | 60 | 3 | 50 |
| **3.** | 823 | 48807205 | 53081647 | 52005 | 0 | 101 | 8 | 102 | 60 | 9000 |
| **4.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **5.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **6.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **7.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **8.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **9.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **10.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **11.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **12.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **13.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **14.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **15.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **16.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **17.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **18.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **19.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

Input Buffer

2 3 4 5 6 7 8.

Output Buffer

listing the corresponding numeric code next to the source. If no errors are found, the resulting numeric form of the program is loaded into VISIAC memory and debugging can commence.

## THE ARCHITECTURE

Reminiscent of the DEC PDP-8, a 1970's era minicomputer, VISIAC's single accumulator architecture operates on words of 52 bits (the largest integer value in JavaScript). Up to 1000 words of memory can be addressed. The instruction set provides 45 distinct instructions. Five arithmetic operations (add, subtract, multiply, divide and modulus) are provided in three variants: from memory, literal value, or stack relative. In each case the second operand and result are the accumulator. A stack register provides an in memory stack usable for both user data and subroutine calling.

The VISIAC character set consists of 100 characters which provide a full set of alphanumerics including upper and lower case and a variety of special characters. The decision to not use a standard character set is deliberate to provoke the discussion of how computers communicate.

In the basic configuration, input-output consists of text input and text output. The IO operations provide for reading and writing single characters or decimal numbers. Additional facilities allow for the status of the input to be tested so that interactive programs can be written. Up to 8 additional IO "devices" may be configured. A device is an independent JavaScript module with a well defined interface. It is envisioned that advanced students could develop their own IO modules. See the side panel for a list of possible IO devices.

To make VISIAC useful in a teaching environment an extensive set of tutorials and exercises are required. Some of these are under development whilst others will hopefully be contributed by the early adopters. It is hoped that these can be used to explain computer science concepts in the context of an easily understood machine architecture. See the box right for further suggestions.

## VISIBLE ACADEMIC COMPUTER TECHNICAL SPECIFICATION

VISIAC is not just another computer simulator. It offers far more than most simple models found online, as can be seen from the specification outlined below.

**Word size:** 52 binary bits, 15.9 decimal digits.

**Memory:** Up to 1000 words.

**Registers:** Accumulator, stack pointer and program counter.

**Instruction set:** 45 distinct instructions including: load, store and 5 arithmetic operations to memory; literal operations; stack push, pop, and relative access; full complement of test and branch operations.

**Execution speed:** Variable 1 to 1000 instructions per second.

**Input-Output:** Text or numeric input output, extensible with up to 9 external I/O devices.

There are many possibilities this opens up. Some of the outcomes could be:

- Switches, lights, 7 segment readouts
- Turtle or plotter
- Karel the Robot
- IO interrupt controller
- Remote terminal adapter
- Inter VISIAC networking
- Disk drive
- Virtual memory controller

## CAN YOU HELP DEVELOPMENT?

Possible example tutorials might focus on:

**Programming language variables**

Explained in terms of memory location, the variable name is the address, the value its contents. Thus the often confusing a = a + 1 can be seen to be lda a; add #1; sta a  when translated into assembler.

**Recursion**

The stack clearly shows how each call preserves information.

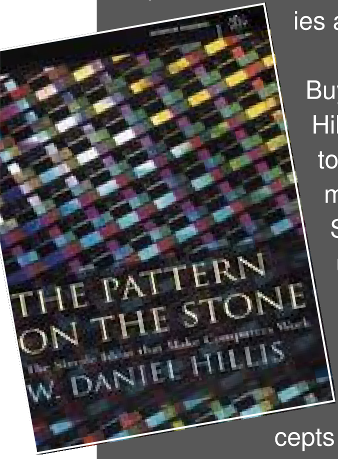
**Scalability**

By examining the possible ways of summing 10 numbers in memory, the concept of the scalability of an algorithm is developed.

If VISIAC seems a useful addition to your teaching aids please consider helping to develop the necessary supporting infrastructure by contacting Hans.Pufal@gmail.com

## THE SIMPLE IDEAS THAT MAKE COMPUTERS WORK

There are few authors in the sphere of computer science who have the ability to present ideas in an accessible fashion like Danny Hillis. Few authors have his pedigree either. Using simple language, metaphor and analogy Hillis wrote 'The Pattern On The Stone' in 1998. Although now out of print, there are many second hand copies available.



Buy one! The clarity of Hillis's prose gets straight to the heart of what makes computers tick. Starting from the bottom up, he provides an outline of basic logic and its implementation in a machine. It introduces all the important concepts in computer science such as finite state machines, programming languages, Turing universality, algorithms and algorithmic complexity. If some of these terms are intimidating, by chapter three you'll feel at home. This is a cracking read that focuses on ideas. Certain concepts recur throughout, in particular the idea of functional abstraction, making computers understandable by decoupling the ideas from the specific technologies.

Once readers are comfortable with these foundations Hillis proceeds to introduce notions such as uncomputable functions, heuristics, parallel computing, neural networks and machine learning. These higher ideas ignite his excitement, yet are all explained in easy to follow terms. He writes, "It is certainly true that a computer can incorporate and manipulate all other media, but the true power of the computer is that it is capable of manipulating not just the expression of ideas but also the ideas themselves… The computer is not just an advanced calculator or camera or paintbrush; rather, it is a device that accelerates and extends our processes of thought." This book allows you to appreciate why computing concepts go far deeper than the information technologies for which they are used. It is, in my view, one of the best introductions to the ideas behind computer science available. *Roger Davies*

## CS4HS: GRANTS FOR UK AND ONLINE TEACHER COURSES

**CS4HS is Google's main annual grant scheme to support Universities in their work with schools. Much activity is focussed on teacher workshops, but there is also limited sponsorship for direct engagement with students and resource creation activities.**

The 2013 EMEA (Europe, Middle East and Africa) round saw 125 applications from 108 universities in 40 countries, resulting in funding for 28 projects from 18 countries. As always, there were many strong applications from the UK and six were successful:

- The University of Kent, who produce Greenfoot as well as running the CAS Online forum system, is receiving funding for delivery of teacher workshops for Computing teachers, both in the UK and internationally, along with the development of an online tutorial video series.
- Queen Mary, University of London, receives a continuation grant in support of their cs4fn magazine, website and activities.
- The University of Manchester is receiving funding to help produce and host online a set of fun, practical, activities so anyone can deliver a workshop on computing based around the Raspberry Pi. The activities will be made available to every teacher in the country (and internationally) so they can excite the next generation with Raspberry Pi.
- The University of Worcester receive funding for a workshop to provide training for teachers specifically in computer programming, including the Python language, Processing (KS3) and C++/Java (A-Level) and creative programming using HTML Canvas and JavaScript.
- Teesside University receive funding for a two day workshop to provide teachers with the foundation computer science knowledge they require to deliver the non-programming elements of secondary computer science in UK schools.
- The University of Warwick is being funded to provide a "supported MOOC" course to give teachers from a wide geographical area the opportunity to learn the new topics they will soon be expected to teach, followed by a workshop for participants.

The Warwick MOOC is just one of several such activities being funded by CS4HS globally this year. For example, the Colorado State University's global campus will offer an eight week problem-based self-paced online workshop to encourage non-CS teachers to integrate computer science concepts into their curricula, and Harvard started a six week online CS course for beginners in June, focussing on Scratch and Computational Thinking. If you have access to a Lego Mindstorms NXT kit, Rowan University's laboratory for educational robotics will offer an online workshop for teachers, in August, that may be of interest. "Start Your Own Robotics Club: LEGO Mindstorms Programming for Absolute Beginners" will help teachers get started in robotics. *Peter Dickman*

## GETTING STARTED RUNNING CPD COURSES

If you are a university department or school within the Network of Excellence and would like to run your own CPD courses for teachers, there is some help in terms of sample course structures and downloadable materials from Anglia Ruskin University available at: http://networkofexcellencecpd.net . *Sue Sentance*

# A NATIONAL PROGRAMME OF PROFESSIONAL DEVELOPMENT

**In June at Holyrood Dr Alasdair Allan MSP announced an additional £200,000 worth of funding per year for the next two years to support us in delivering computing science in Scotland's schools. Mark Tennant and Peter Donaldson explain the details.**

Schools in Scotland are currently undergoing the largest change in a generation to qualifications and assessment, with the implementation of the 'Curriculum for Excellence' now almost complete.  Amongst the new qualifications, Computing Science will be offered at National 3 – 5 (GCSE equivalent), Higher (A level) and Advanced Higher.

The new qualifications put much more emphasis on topics such as computational thinking and problem solving in programming.  Programming-lite options such as Information Systems will no longer be offered, presenting a challenge for many teachers who have little programming experience.

The Scottish Government recognised this and began working with concerned parties – including CAS Scotland – last year.  Dr Allan's announcement has been well received north of the border, and constitutes the largest investment in Computing training since Computing Studies was first introduced in the '80s.

During his speech he made several important points that CAS Scotland members feel haven't always been understood by some of our other teaching colleagues, school leadership teams and parents. In particular, "CfE recognises computing science as both science and technology" and that "For our society and our economy, young people need to be as aware of this new science as they are enthusiastic about using the technologies based upon it."  The full statement is online at http://goo.gl/rjiTt6

We now have a unique opportunity to explore the most up to date thinking on CS pedagogy in the pre-certificate levels and new national qualifications in Computing Science. We also have the chance to create some time and space to think about important issues such as broadening participation, developing computational thinking and improving progression so that our learners don't end up stuck in the shallow end.

At the heart of the proposal put to the government by CAS Scotland are:

- the creation of local teacher communities where a lead teacher can work with others to investigate new practices and techniques in a supportive environment, similar to the Master Teacher programme in England.
- working in partnership with a range of organisations in HE, FE and industry, linking them with lead teachers.
- a flexible professional learning programme primarily focused on the pedagogy of teaching Computing, linked to the new qualifications that teachers can quickly apply.

Teachers' professional development needs are being surveyed by CAS Scotland to provide a detailed picture for the project officers carrying out planning work. Several one day events are being considered to address teachers' immediate needs early in the autumn term. The training of the first group of lead teachers will focus around non-certificate levels and National 4 and 5 qualifications.

During the first phase of the programme CAS Scotland aims to develop 20 lead teachers linked with FE or HE institutions in their area, a customisable programme of professional learning for developing computational thinking in the BGE and National 4/5 Computing Science and delivering this to over 220 teachers across Scotland.

## CAS SCOTLAND TEACHER CONFERENCE OCTOBER 26

Preparations are well underway for this year's Conference. Last year's event in Edinburgh proved a huge success. This year we have moved the venue to Glasgow University so we can accommodate more delegates. Make sure you register quickly. Full details on CompEdNet.

## SWITCHEDON SCOTLAND ISSUE THREE PUBLISHED

The third issue of our CAS Newsletter specifically aimed at members in Scotland, was published before the summer break. The issue is shorter than previous ones but was rushed out to ensure teachers received the wonderful and reassuring news about the new funding proposals for professional development. It includes a call for teachers to get involved in the working group and details of how to access the skills survey.

But that's not all that is covered. Inside you will find a major features on pedagogy and the limits of online tutorials written by Quentin Cutts, which places the classroom as central to effective learning. Charlie Love continues the pedagogy theme by exploring how teachers of Computer Science can make use of flipped classroom techniques.

## A PAUSE FOR THOUGHT

Remember the Y2K Bug? Now Y2K38 looms. The problem stems from applications storing their date/time data in 32-bit form, eg using Unix Timestamp. A timestamp is the number of seconds between a specific date and the Unix Epoch on 1 January 1970. It is essential to systems handling dated information in dynamic and distributed applications, client-side and online.

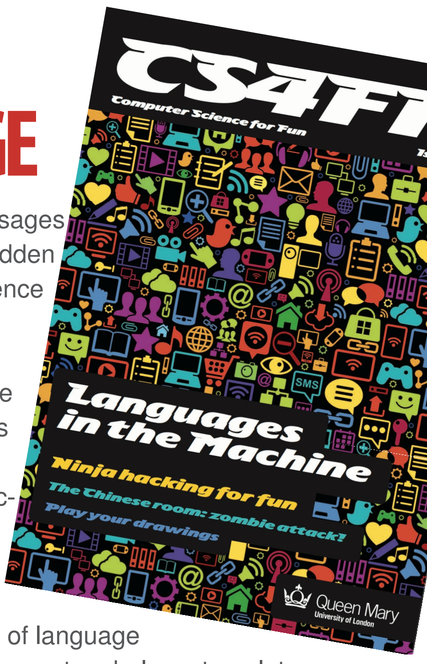
In C, the value type time_t stores time as a 32-bit signed integer, using one bit for the sign, giving us a range from $-2^{31}$ to $2^{31}-1$, or minus 2,147,483,648 to 2,147,483,647. So on the 19 January 2038, when the number of seconds since 00:00:00 01/01/1970 passes 2,147,483,647, the Unix Timestamp will likely fail due to a 32-bit overflow.

Embedded 32-bit processors reside in many systems. All 32-bit systems using Unix Timestamp are affected, unlike 64-bit systems. There is no universal fix, but solutions have been proposed. Systems can switch to unsigned integers, the span then being from 1970 to the year 2106 but unworkable for systems storing dates before 01/01/1970. Perhaps all systems will be 64-bit by then, but some think such an upgrade may not be possible for some existing embedded systems.

Meanwhile, it gives us a useful illustration when teaching our kids about future-proofing. Try for yourself: log out of an online service eg Yahoo Messenger. Now change your system clock to a date after January 2038. Try logging back in to your messenger account: it will likely give you an error and fail to log in.

*Lyndsay Hope*

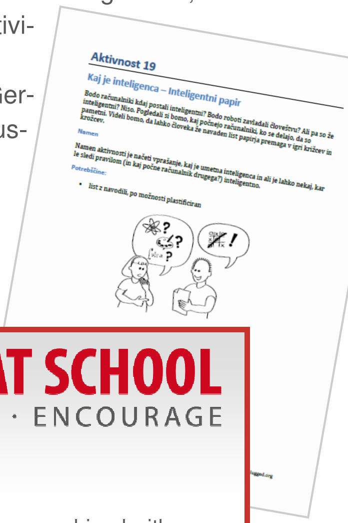
## LOOKING AT LANGUAGE



It may seem odd to the uninitiated but language – that most human of inventions – is also a core topic of computer science. Computers are just machines that can follow instructions. Those instructions have to be written in special languages (programming languages) that are precise and unambiguous. New languages are being invented all the time each designed to make something easy for computers to do. Some make it easier to write programs while others are used to check that computer systems behave the way they should, for example.

Language isn't just the thing that makes computers tick, a lot of exciting computer science goes in to allowing them to do things with human languages. Computer science is inextricably linked with the way we communicate. The Autumn issue of cs4fn is all about computer science and language. We look at how a thought experiment about language has been used to argue that computers can't be truly intelligent and how zombies are attacking that argument. We explain what programming is all about and give you a yummy recipe at the same time. We explore the role language skills play in computer security and how messages can be hidden in the silence of video calls. We look at the limitations of video conferencing and how the umms and ahhs of language matter. Computers help us translate from one language to another; perhaps they may even help to save threatened languages. Language is the essence of computer science.

cs4fn is widely read in UK schools and much admired. But did you know that some of the material has been translated into other languages? Find them at http://www.cs4fn.org/translations/ Over the years, cs4fn magazines, booklets and activities have been translated into German, French, Russian, Italian, Welsh, Portuguese, Greek, Chinese and Slovenian.



Aktivnost 19

Kaj je inteligenca – Inteligentni papir

Bodo računalniki kdaj postali inteligentni? Bodo roboti zavladali človeštvu? Ali pa so že inteligentni? Niso. Pogledali si bomo, kaj počnejo računalniki, ko se delajo, da so pametni. Videli bomo, da lahko človeka že navaden list papirja premaga v igri križcev in krožcev.

Namen

Namen aktivnosti je načeti vprašanje, kaj je umetna inteligenca in ali je lahko nekaj, kar le sledi pravilom (in kaj počne računalnik drugega?) inteligentno.

Potrebščine:

- list z navodili, po možnosti plastificiran

## 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:** *Yasemin Allsop, Phil Bagge, Clive Beale, Pete Bell, Miles Berry, Jonathan Black, Trevor Bragg, Neil Brown, Paul Browning, Mark Clarkson, Paul Curzon, Claire Davenport, Roger Davies, Peter Dickman, Laura Dixon, Peter Donaldson, Mark Dorling, Kate Farrell, Sue Gray, Liz Hadley, Michael Harkness, Graham Hastings, Dawn Hewitson, Lyndsay Hope, Alan O'Donohoe, Adrian Oldknow, Matthew Parry, Hans Pufal, Alex Repenning, Chris Roffey, Sue Sentance, Siraj Shaikh, Andrew Shields, Ben Smith, Chris Swan, Mark Tennant, Jane Waite, Dawn Walker, Arun Warhadpande, Amanda Wilson and John Woollard.*

*Cover image/ primary graphic by kind permission of Krista Shapton (see http://www.kshapton.com/blog)*

**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