THE MAGAZINE FOR COMPUTING & DIGITAL MAKING EDUCATORS

(Hello World)

Issue 5 | Summer Term 2018 | helloworld.cc

DATABASES AND BIG DATA

Why do many struggle with databases - and what can you do about it?

INSPIRING ENGINEERS!

Ideas and inspiration to get your students thinking like engineers

Coding and INCLUSION

Helping visually impaired students realise their potential

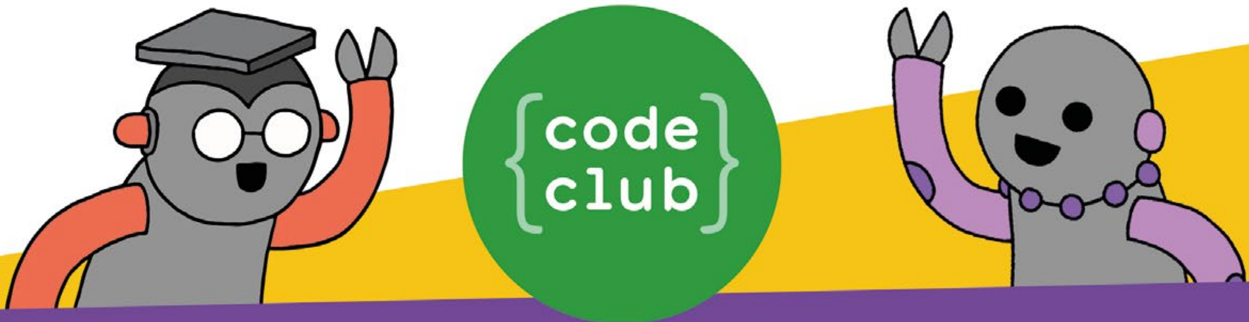
ORGANISING A COMPUTING DAY OUT

Where to go, and what to do when you get there...

PSSST! IS IT OKAY TO COPY?

Why building on existing code is great for teaching

PLUS FLOW CONTROL IN PRIMARY PROGRAMMING • HOW WE BUILT THE RASPBERRY PI • WHY IT'S WORTH TRYING GP! AFTER SCRATCH • MICRO:BIT • TANGIBLE PROGRAMMING IDEAS • 2018: YEAR OF ENGINEERING



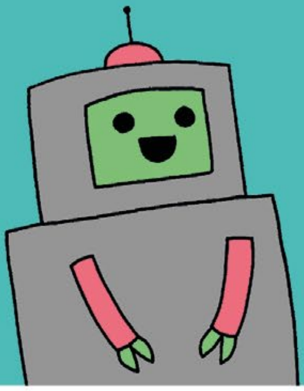
START A CODE CLUB IN YOUR SCHOOL!

Code Club is a network of volunteers and educators who run free coding clubs for young people aged 9-13.

Our aim is to inspire the next generation to get excited about computer science and digital making.

"We use Code Club's fun educational resources to run a weekly after-school club for Year 7 and Year 8 pupils. The students benefit considerably from the extra challenge!"

Karen Dadd, Computing Teacher



- ✓ **Code Club is free**
- ✓ **Code Club provides step-by-step guides for Scratch, Python, HTML, and Sonic Pi**
- ✓ **Code Club helps children develop skills including logical thinking, creativity, and resilience**

We have over 6000 clubs across the UK teaching more than 80,000 young people to code—come and join us!

Find out more at www.codeclub.org.uk

Code Club is part of the Raspberry Pi Foundation. Registered Charity Number 1129409

(Hello World)

CELEBRATING

20,000

···· SUBSCRIBERS ····

SUBSCRIBE IN PRINT FOR FREE

TURN TO PAGE 32

HELLO, WORLD!

2018 marks the Year of Engineering, and so our cover feature for this issue of the magazine focusses on this. We talk of STEM education: at school level, there seems far more focussed on science and maths than technology and engineering, whereas it could be argued that it's engineering and technology that have the greatest direct impact on our lives and our world. Computing is, perhaps, the silent 'C' in STEM, with close links to science (as the zeroth science, and through computational science), mathematics (from Turing and Church's day onwards), engineering (as hardware, and in software development), and technology (for all digital technology). There are some great initiatives now to promote engineering at school level, and perhaps help to develop the 'engineering habits of mind' that Jon Chippindall writes about (p16): check out Maggie Philbin's Teen Tech (p20) and Susan Scurlock's Primary Engineer (p21) for just two examples.

There's a growing number of educators turning their attention now to AI in education. The House of Lords AI Select Committee recently recommended that "children need to be adequately prepared for working with, and using, AI. The ethical design and use of AI should become an integral part of the curriculum." If you're interested in helping the young people you work with learn more about AI, check out Beverly Clarke's new unit of work (p34), and Ken Khan's work bringing AI to block-based coding in Snap!.

We're delighted to welcome many new authors to the Hello World community in this issue. Would you like to join them? If so, pitch us a story idea via **miles@helloworld.cc**.

Miles Berry
Contributing Editor

FEATURED THIS ISSUE

JOHN MALONEY
CREATOR OF GP

John was the lead developer of Scratch for its first 11 years. More recently, he's been creating GP with Jens Monig and Yoshiki Ohshima. He had over 25 years of experience in live, beginner-friendly programming systems.

BEVERLY CLARKE
AUTHOR & EDUCATION CONSULTANT

Beverly is a former teacher and Director of Computing. Now, she is an author and consultant for BCS Chartered Institute for IT. Her latest book, *Computer Science Teacher - Insight Into The Computing Classroom*, is available now.

MAGGIE PHILBIN
BROADCASTER, CEO OF TEENTECH

With over 30 years of radio and television experience, Maggie Philbin OBE has long championed science, technology and medical programming. She co-founded TeenTech in 2008, and is its CEO.

(HW)

EDITORIAL

Publishing Director
Russell Barnes
russell@helloworld.cc

Contributing Editor
Miles Berry
miles@helloworld.cc

PRODUCTION

SBL Ltd
simon@simonbrew.com

Production team
Simon Brew
John Moore
Rachel Storry

Design
Laura Passmore

Cover photography:
BigStockPhoto

CONTRIBUTORS

Critical Media Ltd
Rik Cross
Lucia Florianova
Lucy Hattersley
Nicola King
Oliver Quinlan

Hello World is a joint collaboration:

Raspberry Pi · bcs Academy of Computing · COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE Part of BCS, The Chartered Institute for IT

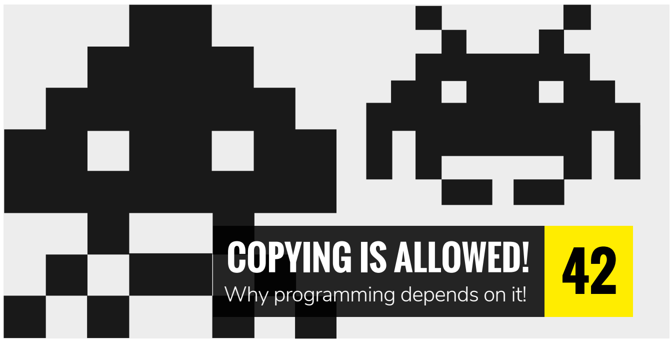
This magazine is printed on paper sourced from sustainable forests and the printer operates an environmental management system which has been assessed as conforming to ISO 14001.

Hello World is published by Raspberry Pi (Trading) Ltd., 30 Station Road, Cambridge, CB1 2JH. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to skills, products or services referred to in the magazine.

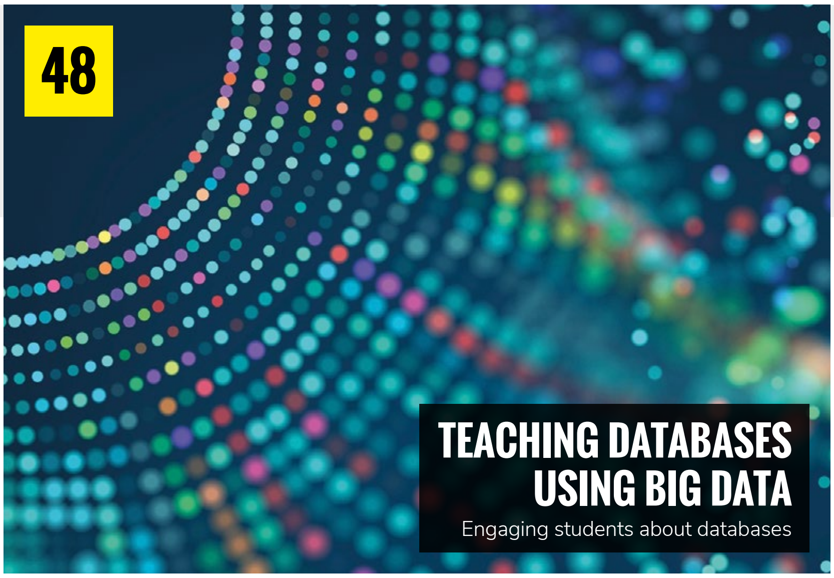
(HW) CONTENTS

ENGINEERING IN FOCUS
Our bumper section starts on page 13!

CODING FOR THE VISUALLY IMPAIRED
The importance of inclusion in the classroom



COPYING IS ALLOWED!
Why programming depends on it!



THE GAMING REVOLUTION
namco

SCRATCH 3.0 DUE IN AUGUST

New version of Scratch to support micro:bit and tablet computers

The MIT Scratch Team is due to launch version 3.0 of its popular educational programming tool this August.

Scratch is a visual programming language that enables students to learn the basics of computer programming by connecting blocks. These can be used to create interactive stories and games.

Set for release in August 2018, Scratch 3.0 will be a complete redesign of the program. The MIT Scratch team aims to implement a number of exciting new features.

"Scratch 3.0 is the next generation of Scratch," say the MIT Scratch team. "With Scratch 3.0, you will be able to play Scratch projects on your phone, create Scratch projects on your tablet, and control Scratch projects with your voice."

Support for tablets (such as Apple iPad and Google Android devices) will help educators working with these increasingly popular devices. You'll also be able to use Scratch 3.0 on Chromebooks.

Scratch 3.0 will ditch Adobe Flash technology (which isn't supported by most tablets) in favour of HTML5.

More information on the technologies being used can be found on the Scratch Wiki (**helloworld.cc/2IdsZyz**).

micro:bit

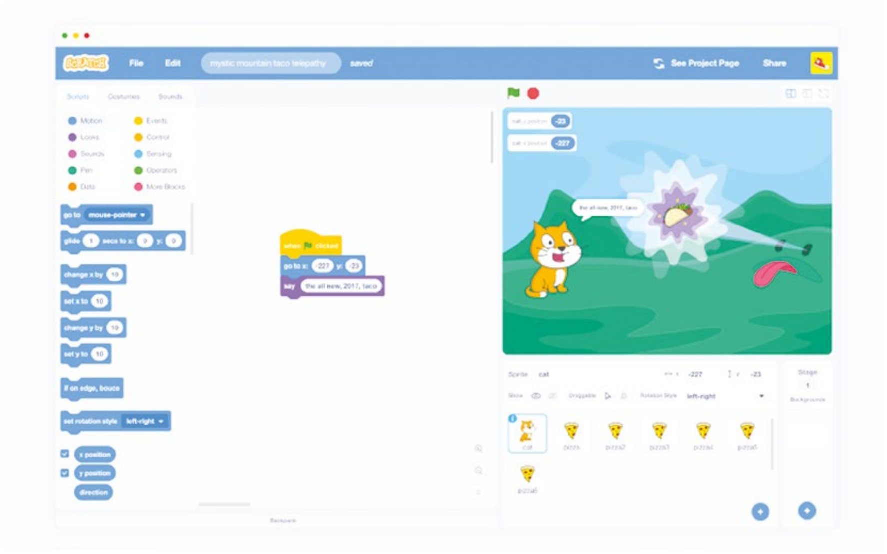
Educators in the UK will also be pleased to hear that the micro:bit is being supported in Scratch 3.0.

A blog post by the team revealed a new extension system with direct support for the micro:bit device (**helloworld.cc/2rj8Axl**). The Raspberry Pi is already supported in Scratch, but sadly other boards, such as the PicoBoard and LEGO WeDo, aren't going to be supported.

The new extension system also opens up Scratch to web services such as Google Translate, which could result in some interesting educational projects.

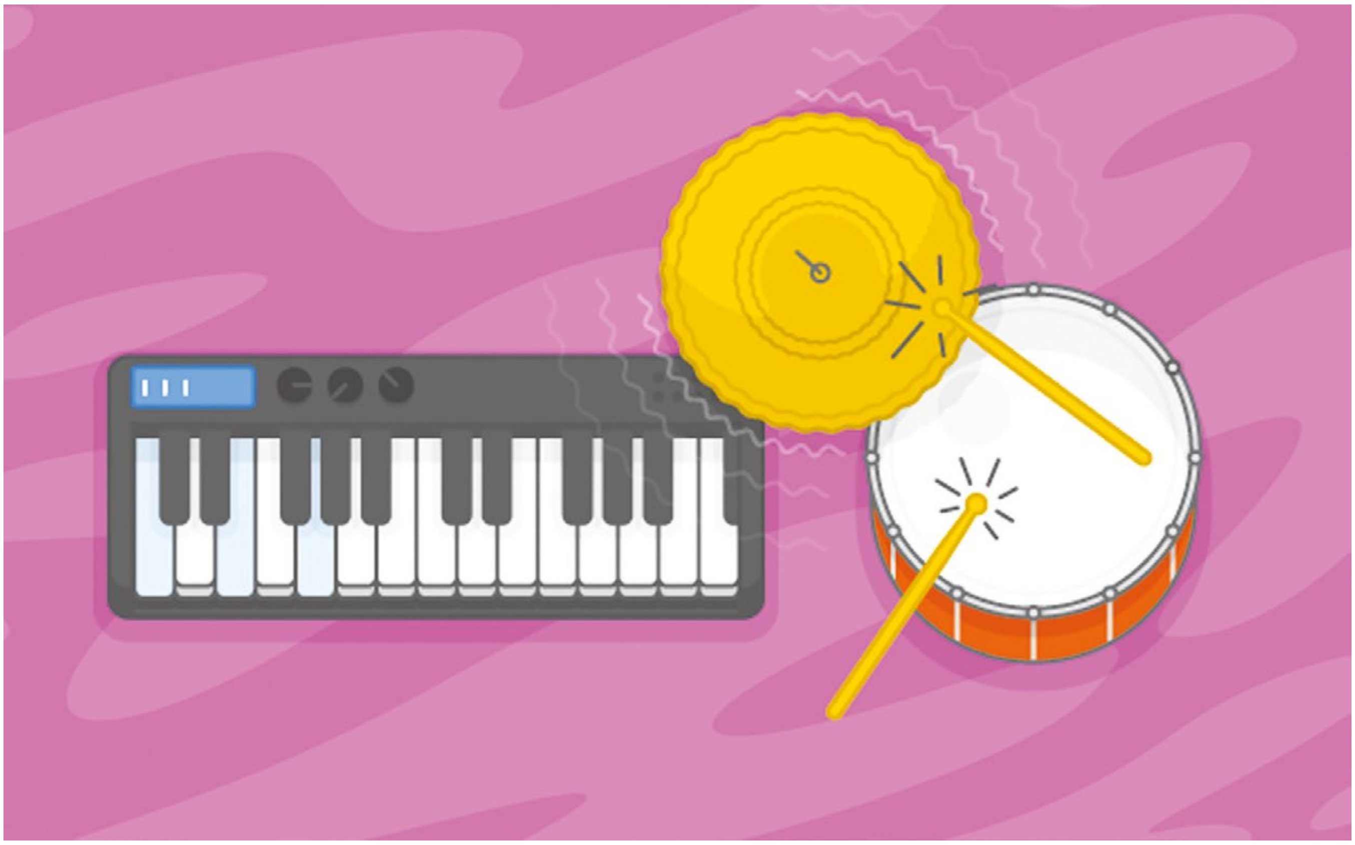
There will also be dozens of new characters, backdrops and sounds available, which will freshen up the program for long-term users. Meanwhile, a new paint and sound editor will enable students to remix and manipulate the characters.

Scratch 3.0 also introduces a range of new blocks, such as sound 'effect',



The new scratch interface is designed for tablet devices such as the Apple iPad. It's larger blocks are easier to tap and drag

"WITH SCRATCH 3.0, YOU'LL BE ABLE TO PLAY SCRATCH PROJECTS ON YOUR PHONE, CREATE SCRATCH PROJECTS ON YOUR TABLET, AND CONTROL SCRATCH PROJECTS WITH YOUR VOICE."



New extensions such as this Music Block take and sound effect blocks are on offer

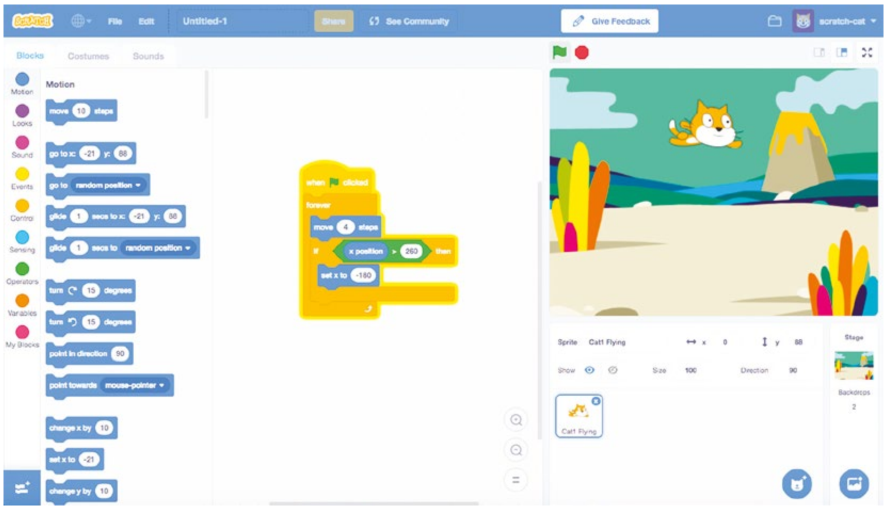
transparency pen blocks and a new glide animation.

Newbie support

The MIT Scratch team has spent a lot of time working on supporting new users. There's a new in-editor tutorial for first-time users and short 'byte-sized' videos introducing concepts.

Teachers will be supported with lots of class activities. There will be new tutorials from Code Club, CS First and updated Scratch Activity Cards and Educator Guides. Plus, an updated Creative Computing Curriculum from Harvard ScratchEd.

A Scratch 3.0 FAQ offers answers to most questions (**helloworld.cc/2HMddve**).



A preview of Scratch 3.0 is available online

Compatibility

All your projects and accounts created in Scratch will continue to work in Scratch 3.0. However, if you've created any lesson plans or educational material, you'll most likely need to rework them to use the new interface.

The visual style in Scratch 3.0 is very different from before, and it has much larger blocks (to make it easier to use on tablet screens). "In most cases, you will want to update your resources to reflect the interface changes. You may also want to create new resources to highlight new features in Scratch 3.0," says the MIT Scratch team.

To help you get a head start on updating your materials, a preview of Scratch 3.0 is already available for you to view and try online at **helloworld.cc/2IdedaW** (HW)

The new AIY kits include a Raspberry Pi Zero WH, so everything you need is in this one box

NEW AIY KITS FOR SCHOOLS

Updated Google Vision and Voice Kits contain all the parts needed to assemble AI

Google has launched updated versions of its successful AIY Vision and AIY Voice Kits.

Made from cardboard and computer components, the AIY kits enable students to build a voice assistant and smart digital camera.

The AIY Projects kits were popular with the maker community, but they also proved to be a huge hit in classroom environments.

"We're seeing continued demand for the kits," explains Billy Rutledge, Google's Director of AIY Projects, "especially from the STEM audience, where parents and teachers alike have found the products to be great tools for the classroom."

Everything included

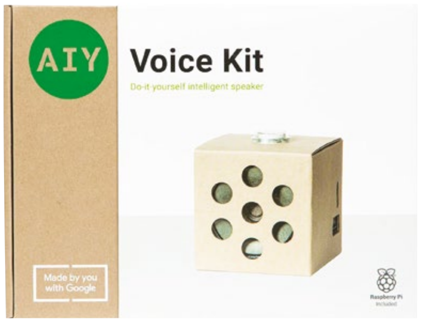
The updated kits include a Raspberry Pi Zero WH, a pre-provisioned microSD card, and – with the Vision Kit V1.2 – a Raspberry Pi Camera Module V2. "Everything you need to get started is right there in the box," says Billy.

The aim is to make setup easier. Users no longer need to buy additional parts or download the software image. Having all the parts in one place makes setup faster, and ensures newcomers have all they need.

The aim of AIY Projects is to make it cheaper and easier to explore AI. "We knew from our research that even though makers are interested in AI, many felt that adding it to their projects was too difficult or required expensive hardware," continues Billy.

The new AIY kits are on sale at US retailer Target. The new kits will be coming to the UK "this summer", Billy concludes.

> THE AIM OF AIY PROJECTS IS TO MAKE IT CHEAPER AND EASIER TO EXPLORE AI



COOLEST PROJECTS COMES TO THE UK

International digital maker show comes to the UK

An international show for young digital makers to demonstrate their projects, Coolest Projects is coming to the UK for the first time this year.

Coolest Projects International will take place at the Royal Dublin Society on Saturday 26 May (**coolestprojects.org**), while Coolest Projects UK is on Saturday 28 April at Here East in the Queen Elizabeth Olympic Park, London (**hereeast.com**).

Anyone under the age of 18, in primary, secondary or further education, can enter as an individual or part of a team of up to five members. Projects are entered into five categories: Scratch, Websites, Games + Web Games, Mobile Apps, and Hardware. The full criteria are at **magpi.cc/2CPqaK5**.

Rosa Langhammer, CoderDojo General Manager, Outreach & Engagement, explains: "Coolest Projects is about bringing an idea and sharing it with your peers, no matter how big or small or even if your project isn't finished." The entry categories are so broad because CoderDojo wants "Coolest Projects to be as inclusive as possible."

There's plenty to do at Coolest Projects, besides demonstrate your own build

Coolest Projects is an international event for young digital makers aged 18 and under

Inspiration across the nations

Coolest Projects started "in 2012 with 19 projects," Rosa confirms. Now it's "an international showcase with 750 young people participating last year from 16 countries!"

Each participant will have their own area to set up their project, and will "have some free time throughout the day to explore other projects, as well as some of the cool demos and speakers that will be joining us for the day," says Rosa.

"It's really important to have hands-on activities scattered throughout," Rosa continues, "so parents, young people, and the general public can get hands-on with science, technology, and the arts."

Rosa's favourite Coolest Project came from 12-year-old Amy, who created a smart beehive "with a mission to save bees!" Amy uploaded data from her hive to **hivetool.net**, helping "international scientists to understand why bees are dying."

CODE CLUB IN WESTMINSTER

Showcase event for MPs and Lords "a special experience"

Code Club's Dan Powell talking with some of the pupils from Coppice Primary School

"TO RAISE AWARENESS WITHIN THE HOUSES OF PARLIAMENT ABOUT WHAT CODE CLUB IS

Code Club came to Westminster in January, to demonstrate how a Code Club works to MPs and Peers. The event was organised by Lloyds Banking Group, hosted by Labour MP Wes Streeting, and Code Club was invited to take part.

Coppice Primary School, from Wes's Ilford North constituency, was invited to "simulate a Code Club in Portcullis House," says Dan Elwick, Code Club regional coordinator, London & East of England.

Code Club participated "to raise awareness within the Houses of Parliament, as well as to staff in Lloyds Banking Group, about what Code Club is, and how it works," Dan explains to us.

"Politicians were coming in during their lunch break," Dan reveals, "and there was a talk from Lloyds Banking Group and Wes Streeting… I think also it was a special experience for the children involved."

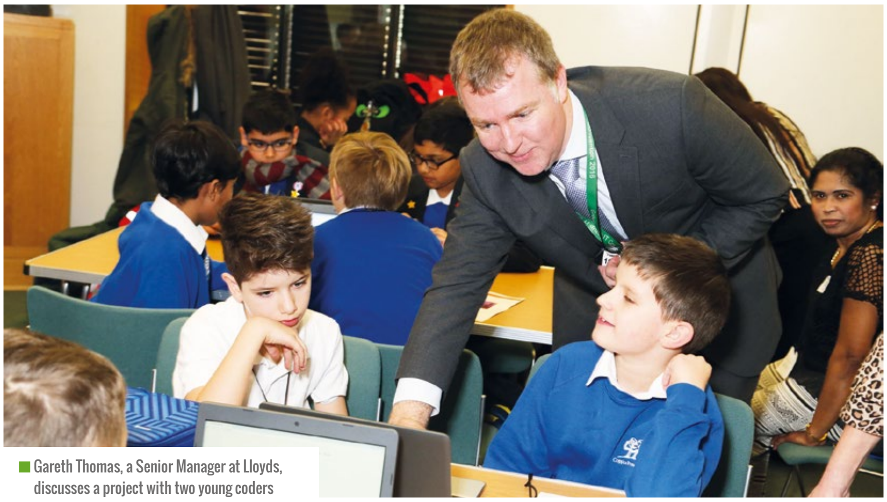
Hosting the event "over the road from the Houses of Parliament", as Dan puts it, gave the children of Coppice Primary the chance to "come to Westminster and experience that environment." It also allowed politicians to see a Code Club in action, and talk to children about their coding skills. "Children were so excited to be there and the MPs clearly enjoyed it too," Dan tells us. "There was a real buzz in the room."

The event ran for an hour and a half, over a lunchtime. In that time, the 28 pupils from Coppice Primary created one of two games, based on Code Club projects.

Playing politics

The children were a "mixed cohort [of] Code Club members and selected pupils from

Pupils appeared to enjoy the pop-up Code Club at Portcullis House in Westminster


Gareth Thomas, a Senior Manager at Lloyds, discusses a project with two young coders

Years 4, 5, and 6", Dan explains. Code Clubs are held for children aged 9-13 years old – see **codeclub.org.uk**.

"So we set the children a couple of Code Club projects," Dan reveals. "There was one fairly basic project, then another more advanced one. All the guests could see the children coding and how much they enjoyed making things with code."

The first project was Ghostbusters, "a game where you clone lots of ghosts, and then 'click' to catch them," says Dan. The second was Clone Wars, a "Space Invaders-style game, where you're shooting lightning bolts at hippos that are falling from the sky." Both projects are on the Code Club Projects page (**magpi.cc/unpyFy**) and use Scratch.

MPs and Peers from Parliament chatted with the children as they worked on their games. "The children found that really interesting," Dan confirms. "I heard one of the Lords saying to the children, 'And that's why you should all stand for election in your local area when you grow up!'" (HW)

CODE CLUB IN LLOYDS BANKS

The event was organised by Lloyds Banking Group, and Code Club was delighted to be asked to take part. As Sarah Sheerman-Chase, Senior Programme Manager, explains, "We've been collaborating with Lloyds Banking Group for about 18 months now, and they're really keen supporters - hundreds of their staff are registered to volunteer with Code Club and their colleagues run over 70 clubs."

Lloyds Banking Group has its "own internal drive for digital inclusion, and Code Club is a key part of that initiative," says Sarah. The scheme is called Digital Champions, where over 27,000 colleagues have pledged to help people or charities with their basic digital skills. It's part of the bank's Helping Britain Prosper Plan, which Lloyds Banking Group says: "takes us beyond business as usual."

Sarah reveals, "There are around 700 [Lloyds Banking Group] colleagues registered with us now", while Lloyds Banking Group "has also started a pilot of Code Clubs in some of its branches."

UKFAST OPENS PI CAFÉS IN MANCHESTER SCHOOLS

Web hosting company UKFast to invest £100,000 in electronic creative spaces

Pi Café
UKFast
UKFast
TELL ME AND I FORGET, TEACH ME AND I MAY REMEMBER, INVOLVE ME AND I LEARN
BENJAMIN FRANKLIN

The five new Pi Cafés will operate much like the pilot site in Broadoak School, Partington

Following a successful pilot in 2015, web hosting firm UKFast has announced five 'Raspberry Pi Cafés' for Manchester schools this year.

The project represents a £100,000 investment from UKFast.

Aaron Saxon, UKFast's Director of Training and Education, reveals, "We are distributing 120 Pis across the five sites: Holy Name RC Primary School in Moss Side, St Bede's Prep School in Hulme, Alderley Edge School for Girls, The Hollins Technology College in Accrington, and The Factory Youth Zone in North Manchester."

The sites were chosen "where gaps in digital engagement exist", Aaron explains. This includes areas lacking "the resources to deliver cutting-edge digital training, as well as all-girl schools which have traditionally seen low uptake in technical subjects."

WE'RE PROVIDING THE TECHNOLOGY FOR THE CHILDREN IN A FUN AND EXCITING WAY

We asked Aaron how the Pi Cafés would actually operate, and it seems that's largely up to the schools: "Some schools may use it as a creative space, others will use it as their computer science classroom as well as an extra-curricular hub and space for the community."

The computers "won't look like traditional desktop units," Aaron tells us, "as we want them to be more computer-science focused." For UKFast, that means "there will be arcade, old-school gaming, and robotics cases" Aaron says. "We're providing the technology for the children in a fun and exciting way."

Paul Grier, Network Manager at St Bede's Prep School (one of the five new sites), adds that "in 20 years' time, 45% of jobs will be done by AI and robots. So if kids today don't understand [these things], they won't understand how the world works."

Paul says he hopes the new Pi Café will "allow both children and the staff [of St Bede's] to delve more into computer science." While students and staff of St Bede's "learn ICT, which is processing and spreadsheets," Paul tells us that "programming hasn't taken off as much as I would have liked it to." (HW)

Rhys Morgan, the director of engineering for the Royal Academy of Engineering, introduces our very special issue...

CELEBRATING ENGINEERING

I'm delighted to be writing a piece for this edition of the magazine as 2018 is the Year of Engineering.

The timing of the 'Year' reflects many causes for celebration, including the fact that 2018 will see the first passenger journeys made on the new Crossrail Elizabeth Line – an enormous feat of multi- and inter-disciplinary engineering. The UK excels at projects like these, demonstrating the quality of project management and the power of visionary leadership in ensuring such projects deliver maximum benefit through innovation, local capacity development, and community engagement.

But it's all too easy to just dive straight into civil engineering and construction projects when thinking about engineering. We rarely consider the contribution that engineering has made in advancements to medicine, sport, entertainment, media, and even retail. This is reflected in images and perceptions of engineers by the public. If you search for the word 'engineer' in Google images, it's amazing to see the consistency of the image, and it ain't great: a sea of men in hard hats and hi-vis jackets.

The IET, a professional engineering institution, recently conducted research into what schoolchildren think a 'typical' engineer looks like. Of a representative sample aged 9-16, 44% imagined they'd wear a hard hat, and 40% thought they'd wear a hi-vis jacket. 67% said that a typical engineer is male, and 51% thought they'd be white. We know from recent research we commissioned that young people are put off engineering by images of hard hats, white coats, and people working alone, whereas images of cutting-edge technology, teamwork, and creativity are much more appealing. And my frustration is that this is much more what engineering is like in the 21st century.

To counter this, as part of the Year of Engineering, we've recently launched a campaign called This is Engineering (**www.thisisengineering.org.uk**). It's targeted at young people aged 13-18 through the social media channels they consume. It's also designed to connect to them through their interests and passions, like music, gaming, sport, and fashion. Since January, we've had over 9 million views of our videos, which show engineers doing a whole range of things young people would never think of as being engineering.

But to get more young people to address future global challenges requires more than just changing the public perception of engineering. In an increasingly dynamic and unpredictable world, they'll need a set of core skills that enable them to take on 21st century challenges. The Academy has identified a set of 'Engineering Habits of Mind' – attributes or characteristics exhibited by practicing engineers and they that provide this ability to adapt. At the core of the engineering mindset is making things.

So in this Year of Engineering, let's celebrate making. Let's get kids making stuff, and get them to see the amazing creativity and excitement on offer through engineering – in its very broadest sense. (HW)

ENGINEERS MAKE THINGS THAT HELP PEOPLE

Katie Henry examines ways to cultivate engineering skill in all students

STORY BY Katie Henry

"Let's get a fan and see if it works," says Edwina. Edwina, a student in my 4th grade class, had just finished building a model of Theo Jansen's Strandbeest – a kinetic sculpture that walks in the wind. As the fan came on, her sculpture began walking across the table, and a few children cheered. I had never seen Edwina smile so big. Then one of the legs fell off. Edwina wrangled it back into place, explaining that she'd need more time to "really fix the leg".

As she and a few other students tinkered with her sculpture, I reflected on what I'd seen: a nine-year-old girl confidently dealing with complexity, persistent in working on difficult problems, flexible, and tolerant of ambiguity. (Others may have noticed that she's dyslexic, reading below grade level, and struggling with a severe speech impediment.) This is a public school.

"You're growing as an engineer," I say. She was puzzled: "What's an engineer?" Max, another student, gave a definition that I still use to this day: "Engineers make things that help people."

Recognising engineering skill

Jennifer Cross, author of *Creative Robotic Systems for Talent-Based Learning* (2017) writes, "Engineering design is the process of developing a concrete solution for an ill-defined problem within technical feasibility constraints." The good news is that engineering doesn't have to begin with costly tools, programming or robotics in your classroom. It's already happening on school buses, at lockers, and – whether you realise it or not – in your classroom. Introducing young people to engineering starts with learning to recognise engineering skill and helping students to recognise it. The next sentence you speak can introduce a young person to engineering. Below are five types of engineering skills, what they might look like in your classroom, and what you can say to help students develop those skills.

- **Intentional design** Look for pre-planning or evidence of thinking ahead.
What you can say: "Tell me more about your planning process." Encourage intentional action and allow students to make their own decisions. Engineers makes things that help people.

- **Innovating** Look for novel or risky efforts.
What you can say: "How did you decide to try it this way?" Try to understand why the student created something new, and help them to consider their efforts from another person's point of view. Engineers makes things that help people.

- **Refining and testing** Look for efforts that repeat and improve each time in order to reach a goal.
What you can say: "Can you share more about your goal?" Encourage the student to focus on their goal, generate more solutions, and consider strengths and weaknesses of each solution. Engineers inspect and adapt.

- **Prototyping** Look for evidence of a student modelling an idea to reach a goal.
What you can say: "In what ways does this model represent your thinking?" Prototyping takes many forms: 'works-like' prototypes are working models; 'looks-like' prototypes are non-working. Engineers use a variety of models, tools, and strategies to better understand their ideas.

- **Communicating design** Look for students sharing ideas about something they're planning or creating.
What you can say: "Who most needs to hear your idea? What would be the best way to share it with them?" Encourage students to communicate their ideas in multiples ways for multiple audiences. Engineers share what they discover and make with others in order to make their ideas better. (HW)

Katie Henry is a former classroom and STEAM teacher, licensed as a school administrator and instructional technology coach. She now works as the Professional Development Manager for BirdBrain Technologies. Follow her on Twitter @KatieHenryLearn

SOFTWARE, ENGINEERING AND GOOD DISCIPLINE

Jason Gorman on how he went from being a hobbyist to a software engineer...

STORY BY Jason Gorman

I started programming in the early 1980s. With just 4K of RAM, my programs were small. A few years later, Dad brought an old IBM 286 home. With a luxurious 512K of RAM and a 20MB hard drive, my hobby programs grew by orders of magnitude.

Like so many, I taught myself from computing magazines and sharing ideas with friends. By my mid-20s, I'd become very proficient at writing buggy software that was difficult to change and didn't really do what the users wanted. It was only by luck that I fell into a nest of software engineers. Until then, I'd been making it up as I went along, doing it the hard way.

It's viewed as vocational, but looking back I realise how much some of these engineering skills would have helped me get the most out of my childhood hobby too. So much time wasted debugging. So much time lost because I couldn't easily get back to a working version of the code. So much time wasted building the wrong features because I hadn't really studied the problem or set out a clear vision. So much time wasted because my code was difficult to understand and change safely.

As a programming newbie, I wish someone had told me about things like use cases, iterating the design, version control, unit testing, what we now call 'clean code' (code that's easy to change), refactoring, and other good 'habits' for programmers.

For sure, software engineering's a wide discipline in its own right, related to – but distinct from – programming and Computer Science. You could devote your entire life to learning it, and barely scratch the surface. And I firmly believe there should be distinct software engineering paths in education from GCSE onwards.

But I also believe the basics aren't rocket science. I could have learned to write simple requirements specifications at age 11. I could have learned to write unit tests. I could have learned to use version control. I could have learned to review code for problems like functions being too long and modules doing too much. I could have learned to do basic refactorings, like extracting code into new functions and renaming modules, safely.

These are practical skills, and any programmer progressing beyond the basics can learn them. They just need to see it being done. That's the experience that transformed me from a hobbyist programmer into a software engineer. After they become proficient with a programming language, why not introduce some basic 'code craft'? It's time those who teach programming and Computer Science got together with those who code for a living and made it happen. (HW)

> "BY MY MID-20s, I'D BECOME VERY PROFICIENT AT WRITING BUGGY SOFTWARE THAT WAS HARD TO CHANGE

Why not introduce some basic code craft?

THINK LIKE AN ENGINEER

The 'habits of mind' of engineers align closely with computational thinking, so why not embrace the Year of Engineering in your classroom?

STORY BY Dr Jon Chippindall

Look around you. How many things can you see created by an engineer? Five? 10? More than 10?! Pretty much everything around you has been 'engineered'. Indeed, the world would be a rather archaic place without the work of engineers, as these entertaining videos by Exxon illustrate: **helloworld.cc/2jl56Hk**

2018 is 'The Year of Engineering' and it's a timely campaign. We need more engineers as a country and, more widely, to tackle the 14 Grand Challenges deemed essential to advancing humanity. Curious about the engineering grand challenges? Learn more here: **helloworld.cc/2JOCL7x**

But what makes a good engineer? How do they think? How do they tackle problems? And where is the overlap here with our Computing and DT education?

EHoMs vs computational thinking

In the Royal Academy of Engineering report 'Thinking Like an Engineer', Lucas *et al* (2014) established that engineers typically draw upon a toolkit of six skills titled 'Engineering Habits of Mind' (EHoMs). **Figure 1** presents the six EHoMs (set within a wider set of learning habits of mind) and is presented alongside the six computational thinking concepts from the Barefoot Computing project (**www.barefootcas.org.uk**). There are clear overlaps here, so let's briefly consider a few of the similarities…

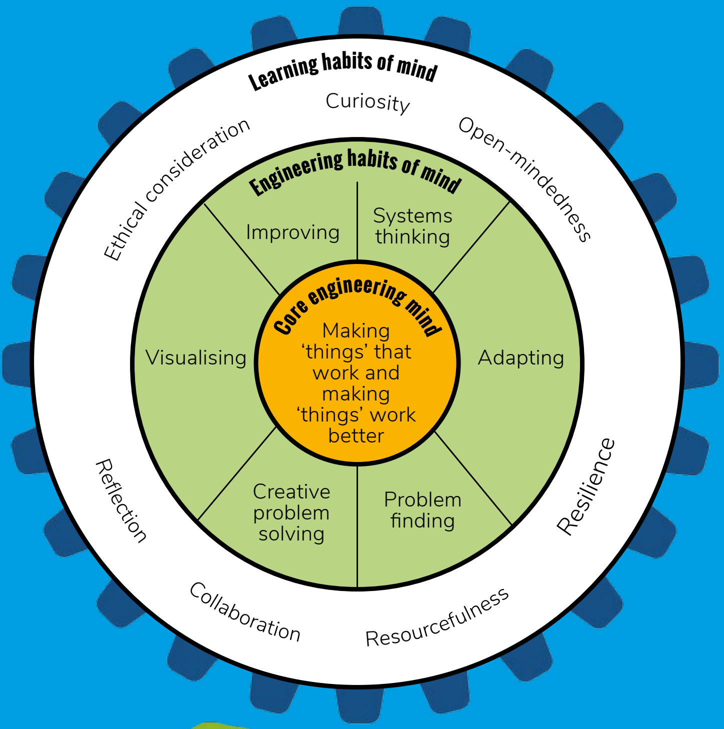
Engineers need to be able to visualise potential solutions to problems, they might make sketches to transfer their thinking to paper in the same way we create plans and algorithms which describe the solutions to our problems. Systems thinking describes engineers' ability to understand how parts of a system interact and work together – for example, how the engine and gearbox in a car combine to drive the wheels. The ability to decompose is a prerequisite to system thinking – how can we break this system down into parts? Likewise, engineers might choose to view a system at different levels of abstraction to manage its complexity. Adapting and improving relies heavily on the ability to evaluate, both during the problem-solving process and afterwards to reflect on how we did. Adopting a logical mindset underpins much of what engineers do, as they apply scientific understanding to creatively problem solve.

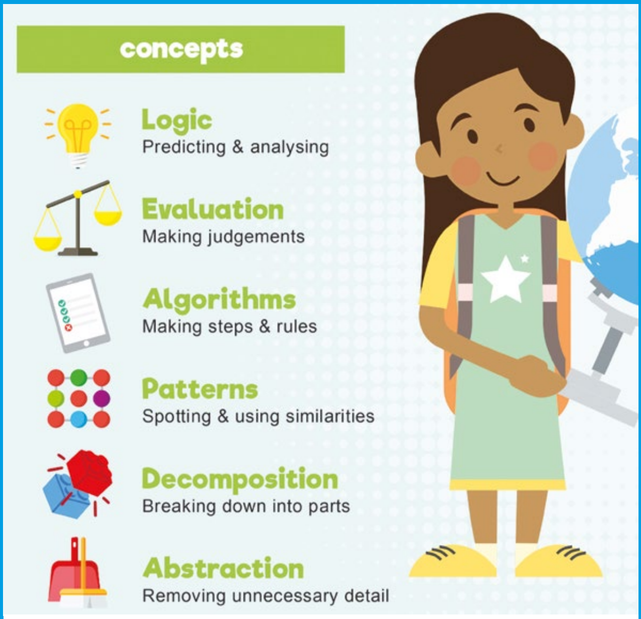
Whilst the short discussion above is far from exhaustive, a clear similarity between cultivating a grasp of EHoMs and computational thinking will help equip

GMEC 2018 – EHoMs IN ACTION

A pupil from Rode Heath Primary School demonstrates their Crumble-powered (**www. redfernelectronics.co.uk/crumble**) marble run they engineered as part of the Greater Manchester Engineering Challenge (GMEC) 2018. Julie Wiskow, Primary Engineering Coordinator from Rode Heath, explained, "Pupils practised their visualisation and systems thinking skills as they envisaged and communicated with one another how the various parts of their marble run would work together." Julie runs her own primary engineering blog at **thinklikeanengineerproject.com**

ENGINEERING HABITS OF MIND VS COMPUTATIONAL THINKING





■ Figure 1: There are close links between the engineering habits of mind and computational thinking, so why not embrace engineering in your classroom to help enhance both? Image: EHoM image from Lucas *et al* (2014)and partial section of www.barefootcas.org.uk Computational Thinker poster

> ENGINEERS TYPICALLY DRAW UPON A TOOLKIT OF SIX SKILLS TITLED 'ENGINEERING HABITS OF MIND' (EHoMs)

pupils with a toolkit of generic and versatile problem-solving skills which they may draw upon to solve whatever unique challenges they encounter.

Give it a go...

The work of SEERIH at The University of Manchester seeks to promote engineering in schools. Their website, **helloworld.cc/2FBtEEy**, hosts a variety of videos developed by in-service teachers collaborating with SEERIH, exemplifying seven principles of engineering education. The website also includes a range of activities aimed at developing each of the EHoMs. Such activities make many cross-curricular links, such as to the physical computing elements of Computer Science. So, since we're in 'The Year of Engineering', why not have a go getting hands-on with engineering challenges in your classroom and start cultivating your pupils' engineering habits of mind? (HW)

Bill Lucas, Janet Hanson, and Guy Claxton (2014), *Thinking like an engineer: implications for the education system*, RAEng. **helloworld.cc/2HOGHUZ**

Jon Chippindall is a primary school teacher and Computing Leader at Crumpsall Lane Primary School. Jon is a CAS Master Teacher, was an author of the Barefoot Computing resources and is a visiting academic at The University of Manchester where he champions engineering for the university's Science Engineering Education Innovation and Research Hub (SEERIH).

HOW WE CAME UP WITH THE RASPBERRY PI

Pete Lomas takes us behind the scenes of the Raspberry Pi, and the challenges the team behind it faced...

STORY BY Pete Lomas

When you've had that great idea, getting it to market is just a bit of engineering, right?

Well, perhaps not. Back in 1995, in an unused portion of an interview for *Triumph of the Nerds*, Steve Jobs famously quoted this as a disease of the then management of Apple, where he felt they thought that the idea was 90% of the work. From my experience with the development of Raspberry Pi, I'm definitely on Steve's side on this.

Beginnings

The idea for the little computer that was to become Raspberry Pi (the name came later) really gelled in a meeting in October 2008 in the William Gate Building by a group that would become founders of the charity.

The original idea was based largely on the BBC Micro concept, booting straight into a Python interpreter when you powered up. It was to interface to redundant (therefore cheap) peripherals that you might have lying around at home, like a PS2 keyboard and mouse with some vanilla wall wart for power.

The display was a bit of agony – we'd all grown up coping with marginal eight-colour graphics on a variety of platforms, but the children of today already had impressive graphics on their PC or gaming console. A 640 x 256 pixel display with eight colours was never going to cut it. Importantly, good graphics capability extends creative reach.

Fortunately, we had someone in the cohort who had access to a chip with HDMI output and a graphics engine to do it justice, codenamed the BCM2727. I got to work with my team on a design, but in reality it fell flat – too many support components and a challenging chip format (BGA with 0.4mm ball pitch) made the design impractical, and we failed to achieve 'Minimum Viable Product'.

Turning point

Scroll forward to 2011, and Eben arrived at a meeting holding a demo of a shiny new BCM2835. This had the critical addition of an ARM core. This had two major impacts. First, the support component count dropped to just two main devices: the memory and a USB hub (that also happened to have Ethernet capability). Second, we could run a variant of Linux, vastly extending the software options, and again increasing the scope of projects that could be tackled.

To validate the potential design, Broadcom kindly supported the creation of the Pi Alpha boards. This gave impetus to projects, with a platform for software development. However, it did pose a large-ish problem: this would cost roughly $110 to make and we needed to come in at $35.

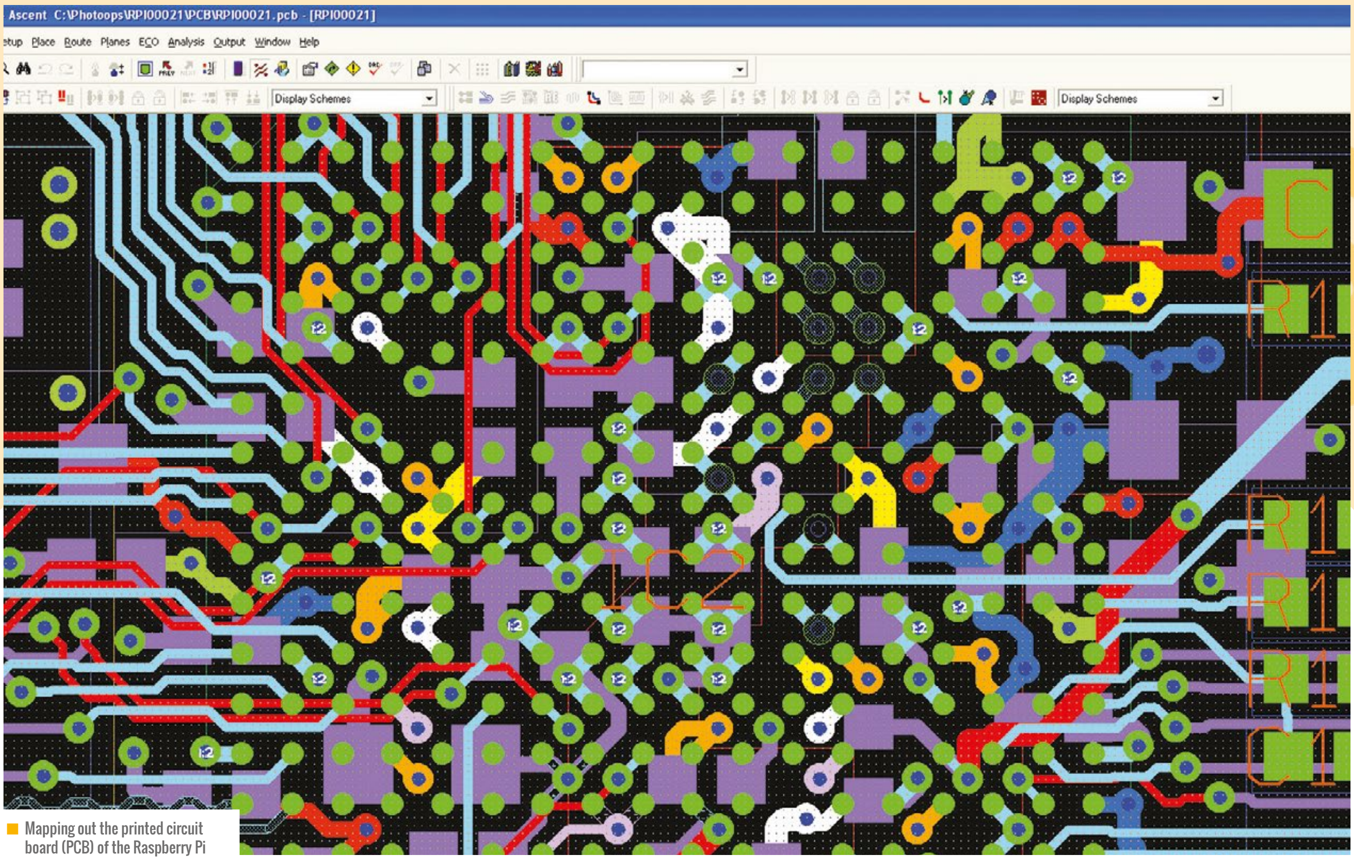
So starting from an ideal specification, we had to attack the engineering problem from several angles: feature set – what the board was actually going to do; detailed design – evaluating the minimal requirement for support circuitry; component cost –

NO SUCH THING AS A STUPID QUESTION

If you look at an early Raspberry Pi, most of the tricky PCB design is under the BCM2835. The Broadcom engineering team had given pretty strict advice on how decoupling capacitors had to be located on the reverse side of the PCB, under the chip.

Normally, I'd use a relatively expensive PCB technology called HDI (High Density Interconnect), but we simply couldn't afford it. HDI consists of a group of techniques, fine tracks, blind and buried vias, and via in Pad, and all good engineering is about intimately understanding the detail (that's where all the best devils live!).

Interrogating my favourite PCB supplier at length - and asking lots and lots of questions - I discovered blind vias by themselves are relatively inexpensive to implement. With a bit of fiddling, I managed to use these to get everything to fit on a simple (and low-cost) six-layer board - phew!


Ascent C:\Photoops\RPI00021\PCB\RPI00021.pcb - [RPI00021]
etup Place Route Planes ECO Analysis Output Window Help
Display Schemes
Display Schemes

Mapping out the printed circuit board (PCB) of the Raspberry Pi

choosing functional but optimally priced components; and the size, structure of the PCB and the placement of components upon it.

Like writing an essay, the first sentence seems the hardest, then it just flows. The months that followed involved lots of detailed design work, looking at options both technically and commercially, and trying to balance the two. In some cases, a feature – for example, a Real Time Clock – would fall by the wayside for the greater good of the price point, but only if it didn't seriously impact the educational mission.

"STARTING FROM AN IDEAL SPECIFICATION, WE HAD TO ATTACK THE ENGINEERING PROBLEM FROM SEVERAL ANGLES

Target

In September 2011, we were so, so close and just needed to push the design over the finishing line, and the cost model under the $35 target. I vividly remember suggesting that if we dropped the Ethernet all would be well. The barrage of abuse (in the nicest possible way) from the rest of the team encouraged me that that would be a serious miss-step. In hindsight, how right they were!

After more late nights over a hot CAD system and numerous telephone chats with Eben, we were there. Prototypes were built in December and, apart from a minor 'is it plugged in?' issue, they all worked.

As any engineer will tell you, never make a fully working prototype – if you do, somebody will ship it! We did, and Raspberry Pi was born. The rest, as they say, is well-documented history! (HW)

Pete Lomas is director of engineering for Norcott Technologies, a UK-based electronics design and manufacturing company. He has spent the last 30 years helping individuals and companies bring their product ideas to life.

He also serves as a trustee of the Raspberry Pi Foundation, whose focus is to put creativity, experimentation and fun back into the teaching of computer science and electronic engineering through digital making. Pete was responsible for the design and manufacture of the original Raspberry Pi hardware.

Pete holds a BSc, MSc. in computer science from the University of Manchester and has lectured on electronic design, computer architecture, systems engineering, and CADCAM. In 2017, he was awarded a DSc. from Manchester Metropolitan University in recognition of his educational outreach activities.

MAGGIE PHILBIN ON THE TEENTECH AWARDS, AND WHO THE REAL STARS ARE...

The entries for the 2018 TeenTech Awards are currently in the hands of our industry judges, tasked with choosing the best 60 projects for the final showcase at the Royal Society. Every single one will receive personal feedback from those experts, whether or not they reach the final, so all students who've taken the first step on the road to being an innovator are helped to see what they can do next.

I read as many as I can mainly because it's such an affirming process. Young people are natural engineers. They enjoy identifying a problem and then finding creative solutions using technology - whether it's how to help a friend with Parkinsons tie up their hair, or how to train your fingers to play difficult pieces on a piano.

I get excited by the bold thinking of the students and by the support teachers have provided. If you're one of those teachers, I want to hug you. We provide industry mentors and run innovation sessions on exciting company and university premises such as GSK, Rolls-Royce, NHS and Atkins to further inspire students, but I know just how much work teachers put into this level of involvement.

I welcome the greater visibility of professional engineers, helping us see they're the game-changers, the ones who'll develop significant breakthroughs across everything from medical technology, to smarter cities, to better hair bands. AI, robotics, data science, or cyber security are the careers of the future. At our events, we bring together the best emerging technology to give young people a sense of what they could be. We show the expected and the unexpected - so at TeenTech City, alongside the very latest tech, they'll also meet Richard The Third (he was 3D printed before being buried) and learn more from University of Leicester about the science and technology that led to him being discovered and identified.

However, the real stars and the most potent influencers are the young people themselves. Over 10 years of TeenTech, I've witnessed what can happen when a young group of students work on a digital project together, and receive encouragement and feedback from our mentors. The confidence of both student and teacher grows exponentially, and the mentors themselves are inspired. One mentor who recently helped a group of students in Brixton said she wanted to cry when she had sight of their submitted project. She could see the difference she'd made, but also was astonished at the quality of the work they'd gone on to produce.

It's humbling to see how the approach is so powerful in reaching a diverse community of students: over 60% are female, and many are on the autistic spectrum, have disabilities or are from challenged social backgrounds. For the past three years, TeenTech students have been chosen to represent the UK internationally as young engineers, and this year a student who'd developed a way to make travel easier for people like herself with narcolepsy was selected.

Over time, schools report significant change. One saw uptake of D&T increase by 300%, another uptake of Physics shoot up to 87.5%. The impact of the approach hasn't been missed by senior management in schools who've not only provided extra support for student projects but in some cases introduced Engineering or Design and Technology departments where none existed before.

I'm just as inspired by the young innovators who share their thinking in their TeenTech Award projects as I have been by the brilliant inventors I met during my years on *Tomorrow's World*. We need to give teachers and schools more space to develop their talent.

Register at **helloworld.cc/2w2w67D** for the 2018/9 TeenTech Award programme (11-16,17-19).

Register at **helloworld.cc/2jlq3BN** for TeenTech City of Tomorrow (KS2/KS3).

PRIMARY ENGINEER
ENGINEERS IN THE MAKING!

Engineering as a subject doesn't exist in primary schools, but used as a way to embed STEM as cross-curricular skills it's the perfect vehicle

STORY BY Dr Susan Scurlock

Aidan helps the Team Proto unveil his design at the Scottish Engineering Leaders Award event in 2016. This is now on display at the Glasgow Science Centre until November 2018

"Miss Brown, I don't like you anymore," said Jade. A little concerned, Miss Brown asked: "Why's that, Jade?" "Because you've been teaching us maths and science without telling us!" STEM by stealth no less! The practical application of skills given a context of engineering has a major impact on children's deep knowledge and also their self-esteem. The engineering habits of mind outlined by the Royal Academy of Engineering and University of Winchester are a skillset that's valuable not only in academic terms but also highly transferable across all areas of education, career paths and professions.

The 'hands-on' nature of engineering encourages visualisation, problem solving, curiosity, creativity, and resilience. When designing the structure of Primary Engineer, we looked to give learning a context and invited engineers to join our training courses to work alongside teachers. More often than not, these courses are offered in industrial settings, enabling teachers to discover engineering on their doorstep and professionals willing to give their time to support whole class project-based learning.

Engineering isn't particularly known as a creative industry, but at its very essence it's highly creative. In 2013, we launched a programme where we asked pupils to interview engineers about their work. Afterwards, the pupils were encouraged to look around them – at home, in school, and in the wider world – and find problems, then design a solution to them. By asking pupils to draw their response, then annotate their drawings and write a letter explaining why their design should be built, we removed the restrictions of budget, technology, and practical skill, focusing on the problem and its solution. The ideas the children have created have been highly innovative, and included in public exhibitions, developed by universities, and put on display in public museums and science centres.

One of the first to be made was the Shopping Trolley for the Elderly, designed by a primary-aged pupil to help his small grandmother lift the shopping out of the trolley and into the boot of her car. Fifth Year Mechanical Engineering students at the University of Strathclyde worked with Aidan, the designer, and us to build a prototype, which was then unveiled at the following year's exhibition. What's been interesting is that we expected engineers to inspire children, but we have found that children can inspire engineers, too! (HW)

Dr Susan Scurlock founded Primary Engineer in 2005 after training as a graphic designer and later as a Secondary School teacher. She developed a sophisticated programme of award-winning teacher training courses, from one-day courses to postgraduate certificates, national competitions and educational programmes involving some of the best-known UK companies, universities and councils. Based at the Burnley HQ with a superb team, Susan travels extensively sharing her Primary Engineers' experience and impact.

#INSIGHTS

DIGITAL MAKING EDUCATORS
A LOOK INTO THE COMMUNITY

STORY BY Oliver Quinlan

Since 2014, the Raspberry Pi Foundation has been building a community of educators committed to bringing digital making to young people. More than 1500 educators have now been through our Picademy training and become Raspberry Pi Certified Educators, creating a community across the UK, North America and the world that can tell us a lot about digital making in schools, libraries, youth clubs, museums, and beyond. Each year, we run an in-depth survey to find out more about the work they're doing, the young people they're reaching, and the challenges they face.

The full report on our website has lots of interesting information about how people are using their training, but here we wanted to share some things we found out that might help other educators to see how their experiences compare with those of others.

Student interests

Educators told us about the interests of their students in different aspects of technology, and not surprisingly they rated this interest very highly. They report that students are more interested in using technology generally and being creative with technology than specific skills like programming or physical computing. We think this shows the importance of framing the broad benefits of learning skills like programming, allowing students to work on using programming to solve problems they care about rather than teaching it as an abstract set of skills.

Qualifications

Only a third of the educators surveyed had a degree in computing, computer science or a similar subject. This is relatively low amongst a group of people who are providing lots of innovative experiences of computing and digital making to the young people they work with. Although technical knowledge is useful, educators who don't have a degree in the subject should feel confident that they can also create similar opportunities for young people.

How are people teaching?

The most popular approach to teaching computing was visual programming, with 81% of educators surveyed using this compared to 72% using text-based programming. Physical computing was also very popular, with 73% of those surveyed using this approach, but only 1% reported using unplugged approaches to computing. Everyone has different areas they're confident in. While nearly 90% of educators we surveyed said they felt confident teaching programming, the more specific areas we asked about like visual or text-based programming had less confidence reported. Computing is a big area, and educators should feel okay that they're not confident in every area of it – many are still learning.

Languages and tools

We asked about programming languages and tools used, and it wasn't surprising to see Scratch and Python dominating, with over 80% of educators using each of them. There's a long tail of languages though, with educators using everything from HTML/CSS and JavaScript to C++, VisualBasic and even languages focused on creative and arts applications like Processing. There's lots more information about languages and IDEs used by teachers in the full report.

Equipment challenges

One of the biggest challenges people reported was getting access to the equipment they need to cover different

aspects of computing. Although it might sometimes look like everyone but you has access to the best equipment, many people are making do with small numbers of computers, Raspberry Pis or resources for physical computing. Some educators in our network have been using crowdfunding initiatives such as RocketFund to generate funding for more equipment. Others are innovating in the way they teach, and rotating small groups around specialist equipment so that they can all cover physical computing.

Finding time

Another big challenge for educators is time, particularly the time it takes for them to learn new skills that they can pass on to their students. Even those who've been on training courses feel they need more time to focus on particular skills like mastering Python or Scratch. To support this, we've created a range of online learning courses focusing on particular skills educators need. Over a four-week course of a few hours per week, educators can further develop their skills, and share ideas and practise with other educators. There are lots of courses available at **helloworld.cc/2I9qLQQ**.

> “ WE'VE CREATED A RANGE OF ONLINE LEARNING COURSES FOCUSING ON PARTICULAR SKILLS EDUCATORS NEED

Sharing the learning

It's really interesting to see that people who've been on Raspberry Pi training are sharing what they've learned with many others. Seventy-five percent have passed on their learning, but our survey seems to show they often focus on doing this at events such as TeachMeets and conferences outside of their organisation.

We know that teachers sharing learning with their immediate colleagues through staff meetings, coaching, or planning together can have some of the biggest impact. We'd encourage educators who've had training on computing (or any subject area) to make a plan to share what they learn with their own colleagues. As well as spreading the impact of the training, it gives you an opportunity to revisit and embed what you've learned. Planning with colleagues soon after training makes you commit to putting things into practice, before the busyness of the day-to-day routines take over again.

We found out so much from the educators we surveyed – if you were one of them, thanks for your feedback. At the Raspberry Pi Foundation, we're using this to shape the work we do with educators, but we hope it's useful for others to find out what's happening in the community.

You can read more from our survey of Raspberry Pi Certified Educators, as well as other research carried out by the Foundation, at **rpf.io/research**. (HW)

COMPUTATIONAL THINKING, CONFIDENCE AND PERSONALITY

STORY BY Lucia Flóriánová

What's computational thinking?

Computational thinking is a topic widely discussed within computing education. It's considered to be a key thinking skill that students should develop in order to live in the technology-driven society.

Logic and problem solving

Computational thinking is generally understood as a type of logical thinking that can help students problem solve; both with and without using computing devices. It's, therefore, seen to relate to children's cognitive abilities, such as reasoning, spatial, numeracy, or problem-solving ability.

More than cognitive ability

Roman-González and his colleagues, however, found out that this isn't necessarily the right view. They studied how computational thinking is linked to non-cognitive factors – self-efficacy and personality. Quantitative analysis suggests that only 27% of computational thinking is explained by cognitive factors. Almost the same proportion – 24% – is related to non-cognitive factors, mainly self-efficacy, openness to experience, conscientiousness, and, surprisingly, extraversion.

Self-efficacy

The study has found a possible link between students' computational thinking and self-efficacy – students' perception of how well they performed. It appears that if learners believe that they can perform well, they're more likely to achieve better results. Lower girls' self-efficacy could also partly explain the gender gap in computing.

Openness to experience and conscientiousness

Computational thinking also seems to be linked to openness and consciousness. This finding doesn't come as a surprise; these are the aspects of personality that are most closely related to academic performance, dealing with open-ended problems and persistence in working on difficult tasks.

Extraversion

A positive link between computational thinking and extraversion, on the other hand, contradicts most existing research. It also challenges stereotypes about computing. Although the consensus used to be that brilliant programmers are usually introverted, this seems to be changing. Computing is becoming less of a solitary activity, and involves more social interaction and collaboration.

Improving computational thinking

These findings shine a fresh light on how we should approach computing education. Although computational thinking is a cognitive psychological process, it's significantly influenced by non-cognitive factors. It's thus paramount to teach computing in an environment that welcomes different types of personalities and is freed from stereotypes about a 'typical' computational thinker.

To make children think computationally, it's important to focus on their development beyond cognitive abilities, such as boosting confidence in programming. It would be especially useful to improve children's self-efficacy in specific computing tasks. Making children see their success through concrete examples can be one way of making them learn more.

Read the original article at **helloworld.cc/2HJZxkl** (Article is behind a paywall.) (HW)

PROGRAMMING PEDAGOGIES: PRIMM

STORY BY Oliver Quinlan

As more and more young people are starting to learn programming, there's a need to develop the pedagogies we use for teaching it. Teaching programming is challenging: as an educator, you can know the subject well yourself, but often find students get exasperated with mastering what you've taught them.

Dr Sue Sentance and the Computing Education team at King's College London found that there was a need for specific strategies educators can use to support the development of understanding in computing. Building on the literature suggesting students need to develop accurate code tracing before they can become successful in programming, they developed the 'PRIMM' model as a structure for activities.

PRIMM stands for Predict, Run, Investigate, Modify, Make. It has been developed by building on the Use-Modify-Create model from Irene Lee and colleagues. The model supports educators to design experiences where students engage with code in a progressive variety of ways that build their understanding.

> STUDENTS ENGAGE WITH CODE IN A PROGRESSIVE VARIETY OF WAYS THAT BUILD THEIR UNDERSTANDING

Predict Students start with the code for a program that works, and are asked to look at it and predict what it does.

Run They run the code and observe what the program does, testing their predictions.

Investigate They then get into the details, looking at the program line by line, and exploring how it does what they predicted and then observed when they ran it.

Modify Students use the understanding they've developed of the code to change it to do different things.

Make Using the concepts in the program that they've now understood and adapted, students make a new program that solves a different problem.

This approach takes the evidence-based approach of starting with existing code, and building the students capacity for code tracing before asking them to start writing code. It scaffolds their developing understanding in a way that teachers can repeat for working with different constructs and programming approaches. The team at King's have trialled this approach with teachers and seen some promising results. They're currently undertaking a wider scale study into the approach with secondary teachers in the UK.

You can read more about PRIMM and the ongoing research into this pedagogy on the Computer Science Education at King's blog at **helloworld.cc/2rh3yCI** (HW)

GREG MICHAELSON PROFESSOR OF COMPUTER SCIENCE

FUNCTION FOLLOWS FORM

Recursion can feel less scary if we focus on information structures rather than algorithms

People who first learn to program using iterative constructs like WHILE and FOR often find it hard to get their heads round recursion. It's commonly covered late in the syllabus, and motivation isn't helped by pointless arithmetic examples like Fibonnaci and factorial. Mutterings about functional programming, and strange concepts like anonymous and higher order functions, add to the aura of mystique.I think recursion is pretty straightforward, but, like any technique, it makes most sense in an appropriate context.

In general, 'function follows form' (riffing on the modernist slogan of 'form follows function') is a good guide for problem solving; that is, the structure of a program should reflect the structure of the data it processes. Or, in computational thinking terms, the patterns in the data should lead to corresponding abstractions in information structures and algorithms.

Recursive structures

Recursion is particularly useful for manipulating linked information structures, like lists and trees. These naturally lend themselves to recursive descriptions and hence to recursive algorithms.

In general, for a recursive structure we can distinguish:

- A **base case**, where the structure is empty

- A **recursion case**, where the structure has values and recursive components.

For example:

- A **list** is empty or is a cell with some value and the rest of the **list**

- A **tree** is empty or is a node with some value and left and right branches which are **trees.**

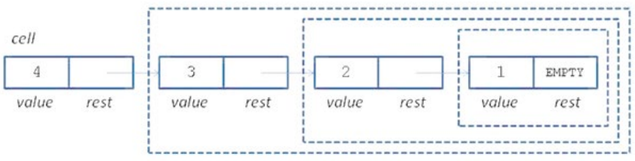


Figure 1. Lists are made of lists

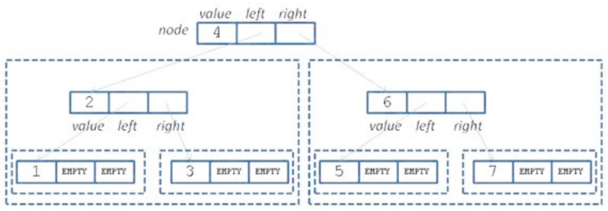


Figure 2. Trees are made of trees

In both cases, we can see the stereotypical notion of a recursive structure being defined in terms of itself. However, I think it's more helpful to think of recursive structures being made up of components that are the same shape as the whole structure: a list is made of lists and a tree is made of trees. See **Figures 1** and **2**.

Recursive algorithms from recursive structures

Once we've chosen a recursive structure, it can be processed by a recursive algorithm that traverses it by following the shape. Typically, we construct:

- A **base case**, where we return some final value

- A **recursion case**, where we process the recursive components and combine the results with the current values.

For example, to sum the values of a list of integers:

- If the list is empty return 0

- If the list isn't empty, add the cell value to the sum from the rest of the list (**Figure 3**).

For example, to sum the values in a tree of integers:

- If the tree is empty, return 0

- If the tree isn't empty, add the node value to the sum from the right branch and the sum from the left branch (**Figure 4**).

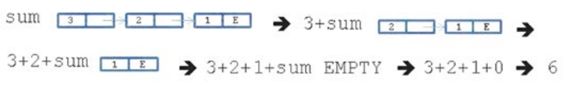

sum 3 2 1 E → 3+sum 2 1 E →
3+2+sum 1 E → 3+2+1+sum EMPTY → 3+2+1+0 → 6

Figure 3. Summing a list

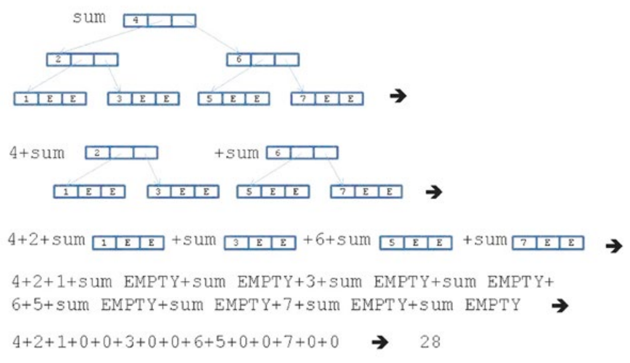

sum
4+sum
+sum
4+2+sum +sum +6+sum +sum
4+2+1+sum EMPTY+sum EMPTY+3+sum EMPTY+sum EMPTY+
6+5+sum EMPTY+sum EMPTY+7+sum EMPTY+sum EMPTY
4+2+1+0+0+3+0+0+6+5+0+0+7+0+0
28

Figure 4. Summing a tree

Recursive arrays

We can also develop recursive algorithms for structures that we'd normally think of as linear. For example, we commonly treat arrays as linear and traverse them with loops, but we can rethink them as recursive:

- An ascending index array is empty or has a first element followed by the rest of the array (**Figure 5**).

In general, to process an array according to this recursive definition, we need to keep track of the index of the first element. If the index is the same as the array length, there's no more array, so we've hit the base case and the recursion stops. Otherwise, in the recursion case, we increment the index to get to the first element of the rest of the array.

For example, to sum the elements of an array, we can modify the linked list recursive algorithm:

- If the array is empty, return 0
- If the array isn't empty, add the first element to the sum from the rest of the array.

Suppose the array is of length n and the first element is at index 0. We'll use a formal parameter `i` to keep track of the index of the current element and increment it each time we move along the array. We'll know we've run out of elements when `i` is `n`.

Coding in C:

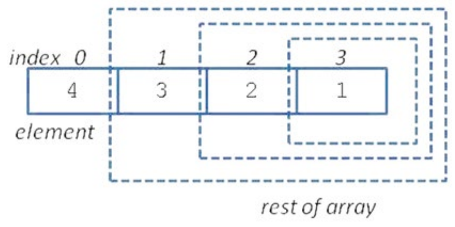


Figure 5. Arrays are made of arrays

```
int sum(int a[],int i,int n)
{ if(i==n)
  return 0;
  return a[i]+sum(a,i+1,n);
}
```

Then, starting with array `a == [3,2,1]`:

```
sum(a,0,3)➔a[0]+sum(a,1,3)➔a[0]+a[1]+sum(a,2,3)
➔a[0]+a[1]+a[2]+sum(a,3,3)➔a[0]+a[1]+a[2]+0➔
 3+2+1+0 ➔ 6
```

It looks like we've reinvented the FOR loop.

Let's try searching an array for value `v`. We'll return the index, or -1 if we can't find the value:

- If the array is empty, return -1
- If `v` is the first element, return the index of the first element
- Otherwise, return the index of `v` in the rest of the array.

```
int search(int v,int a[],int i,int n)
{ if(i==n)
  return -1;
  if(a[i]==v)
  return i;
  return search(v,a,i+1,n);
}
```

So:

```
search(1,a,0,3)search(1,a,1,3)search(1,a,2,3)2
```

and:

```
search(4,a,0,3)search(4,a,1,3)search(4,a,2,3)
search(4,a,3,3) -1
```

Do we need recursion?

Arguably, arrays are intrinsically linear and so should be processed by iteration. After all, Von Neumann machine memory is linear and manipulated by iterative loops built from tests and jumps. Recursive code must be compiled down to such loops, typically treating memory as a stack. Indeed, we could just forget about recursion and always write iterative algorithms. However, realising apparently simple algorithms to process recursive structures can be tortuous.

I think that at the start of any problem, rather than worrying about the algorithm, we should focus on choosing the right information structure for our data. Then, that will be a good guide to how we should structure our algorithm. And if we can see different information structures in our data, we should follow Occam's Razor and choose whichever makes our life easiest. (HW)

CS FOR ALL. BUT ARE WE LACKING VISION?

The challenges facing young, visually impaired coders, and the importance of inclusion in clubs and the classroom

STORY BY Steve Barton

Kids love clubs. Scouts, Brownies, swimming, athletics; they all offer that sense of belonging. A shared interest. A common pursuit. Social inclusion.

Keen to find our eight-year-old son a suitable extracurricular activity, we very quickly exhausted all the usual options; Archie isn't sporty and wasn't realistically going to make the team at our local football club. And he lasted just one Sunday morning taster session at Stage School, brutally crushing his mother's dreams of one day walking the red carpet at some glitzy West End opening night. Fortunately, she did get a full refund on her frock.

But Archie likes to code. That's right – he likes to code. As educators and IT professionals, we promote the educational benefits of coding, but let's never forget; for some, as alien as it might sound to others, coding is an enjoyable hobby.

The T.E.D project uses just the arrow keys to navigate

Code Club member working with the T.E.D project

And that's how I first became involved in Code Club. With no coding clubs operating locally, I approached Archie's school and offered my services as a coding mentor if they would agree to host a club. Fantastically, they enthusiastically embraced the idea and, with the support of The Code Club network, within a few weeks Deputy Head Mr Fry and myself were running our first meeting – full of enthusiastic, likewise children. A kind of Cub Scout meeting for geeks. In a very cool way.

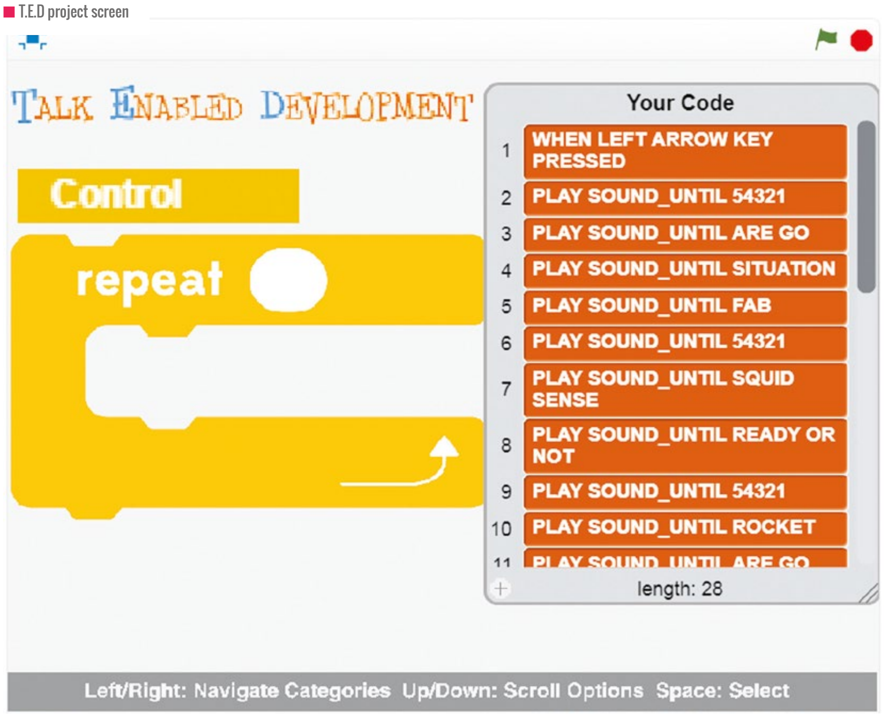
Programming blind

But week two threw a curveball when Ted joined the club. Ted is registered legally blind. We had access to a Big Trak programmable tank, which occupied him for a while. But come week three, as the other children sat in front of their computers creating imaginative, interactive Scratch projects whilst Ted just tinkered with the Big Trak in a space we'd cleared for him on the floor, I couldn't help but think that this wasn't the inclusive club that I had set out to create. We had to do something more.

But what? And how? How does someone who is blind even code? I've been coding for around 25 years and, despite the occasional outrageous brag to the contrary, I really can't do it with my eyes closed. But then I came across an excellent article "A Vision of Coding, Without Opening your Eyes" written by Florian Beijers, a coder from Arnhem in the Netherlands. In his enlightening piece, Florian, blind since birth, explains how all he needs to code (or access any other apps for that matter) is a screen reader, which does exactly what

The T.E.D project uses voice prompts

T.E.D project screen



Your Code

```
WHEN LEFT ARROW KEY PRESSED
PLAY SOUND_UNTIL 54321
PLAY SOUND_UNTIL ARE GO
PLAY SOUND_UNTIL SITUATION
PLAY SOUND_UNTIL FAB
PLAY SOUND_UNTIL 54321
PLAY SOUND_UNTIL SQUID SENSE
PLAY SOUND_UNTIL READY OR NOT
PLAY SOUND_UNTIL 54321
PLAY SOUND_UNTIL ROCKET
PLAY SOUND_UNTIL ARE GO
```

length: 28

Left/Right: Navigate Categories Up/Down: Scroll Options Space: Select

it says on the tin and reads the text on the screen back to him, in a kind of Siri fashion.

I contacted Florian and, through various exchanges, learnt a lot from him. Mainly that he's pretty smart. Florian, or Zersiax as he's known in coding circles, learnt to code at just 10 years old, almost by accident, when he stumbled across an online HTML tutorial written by an older, also blind, student. Using his screen reader, he followed the tutorial and taught himself to code.

See; I told you he was smart.

But scarily, in coding terms, 10 years old is now practically ancient. The UK curriculum is asking children aged just six or seven to learn the rudiments of coding. This is of course where Scratch and other 'blocky' code languages come into their own. But here's the snag; screen readers only read text. The colourful drop and drag code blocks of Scratch aren't text, but images, and a screen reader simply can't work in this instance. As Zersiax explains, "This makes getting started for a kid in this age group trickier than it should be."

But even with screen reader capabilities, there's a further challenge: "You need to find some way of giving these kids that same instant feedback sighted people get with Scratch almost morphing in front of their eyes when they build things," adds Zersiax.

Lack of development tools

This is an issue. Currently, we're expecting children like Ted to follow the Computer Science (CS) curriculum and learn to code without making equivalent, engaging and stimulating development tools available to them as we do for sighted children. And we're putting them at a completely unnecessary disadvantage.

Let's put things into some perspective – figures published by the RNIB estimate the number of children aged 16 and under with vision impairment in the UK and England is 26,000. That's 26,000 school kids who aren't receiving the same level of support in Computer Science as sighted kids to meet the expectations set in the National Curriculum. How's that fair?

What's so disappointing about this is that visual impairment or blindness isn't an obstacle to becoming a great coder (I cite here the super smart Zersiax as one such example). The new CS curriculum was introduced in part to tackle an industry skill shortage, and yet it's inadvertently eliminating 26,000 potential new coders. And coding isn't a bad job for anyone.

Computer science for all

But moreover, and putting education and career aspirations to one side, we're excluding children from participating in a fun, creative and engaging activity that other kids can just take for granted. If we're truly to embrace 'CS for All', and we should, then this needs addressing.

So, what about Ted? What Scratch does have is an excellent sound library. To engage and include him in the Code Club, I wondered if it would be possible to develop something using Scratch to leverage that sound library along with some of the key code categories (Events, Control, etc.). The goal was to give Ted a real coding experience, the same as everyone else. Of course it had to have voice prompts and use the keyboard as little as possible; just the arrow keys to navigate and the space bar to select an option.

The result was a somewhat rudimentary prototype Scratch project I eponymously dubbed 'Talk Enabled Development' (or T.E.D for short). It's basic for sure, but importantly Ted is now enjoying coding his own projects and sharing them with the rest of the club.
Mission accomplished! (HW)

STATISTICS

- Seven in 10 children and young people with vision impairment attend mainstream schools
- The educational attainment of pupils with vision impairment up to the age of 16 isn't as good as pupils with no special educational needs and/or disabilities (SEND)
- Young people with vision impairment are twice as likely (44%) as their sighted peers not to be in employment, education or training (NEET)

Steve is a volunteer Code Club Leader at St Michael's C of E School in Farnsfield, Nottinghamshire, and founder of Eye Can Code, a charity providing support and resources to get visually impaired children coding.

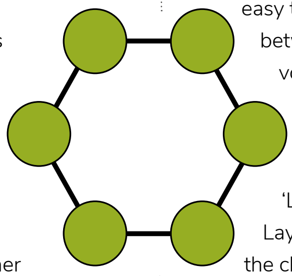
THE POWER OF NETWORKING

For our young learners to better understand the internet, we need to follow the development of networking from the first connected computers

STORY BY Nicholas Provenzano

The power of a single computer is well understood. The speed and accuracy with which it can run through an algorithm and solve the most complex of problems is outstanding. However, it's the mere tip of the iceberg when compared to the power of networked computers.

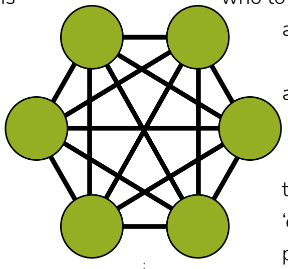
The internet is the biggest worldwide network of devices and has truly transformed almost every aspect of how we live, work and play. Virtually all our infrastructure is reliant on this global connectivity. The ubiquitous nature of its availability together with ease of access has resulted in it replacing costly dedicated connections and transforming everything. Electrical generation, water distribution, transport networks, banks, government, the media, and most of education would all cease to function with no network connectivity. Children can relate to the loss of Facebook or Snapchat, but fail to realise there would be no power distribution and little food in the shops if there was no network.



When trying to explain how the internet works, it's vital to appreciate that it has been a journey over some six or seven decades, with a variety of technologies coming and going. Thus, the networks we have today and the rules they follow are partially a legacy of older technology. If we had a clean slate and could start again, we wouldn't build the internet we have today. Hence, to understand the internet, we need to follow the development of networking from the first connected computers. Please note this isn't a historically authoritative account.

Early connections

As computers developed, it was recognised that they could be interconnected to share information. With two computers, it was easy to provide a dedicated link between them and use electrical voltages to represent the binary 1s and 0s of the data to be sent. The electrical signals are referred to as 'Layer-1' or 'Physical Layer' because they're the closest to the physical connections.



Three computers can be connected with three connections. Four computers need six connections, and so on. The advantage of the system was that each computer could choose where to send the information just by selecting the appropriate connection. The downside was the number of connections and thus connectors on each device. What was needed was a way of using a single wire that all devices could connect to and a way to somehow share usage of the wire between them all. Two distinct solutions evolved – the 'ring' and 'bus' topologies.

Ring

Rather than connect each computer directly to every other computer, they can be connected in a ring topology. The data passes round the ring, with each computer 'seeing' the data but also passing it on. The computer that puts the data in the ring can remove it if it returns. However, how do we ensure the data only 'goes to' the computer we want to send it to, since it clearly now goes through all computers?

The solution involves giving each computer in our ring a unique address. Before we send our data out, we add the destination address on to the front of the data. For the receiving computer to know who to reply to, we also add our own address as the source address.

This addition of a source address and destination address to the data forms a 'frame'. The process of adding this additional data is called 'encapsulation'. It's similar to placing the data in an envelope, putting the address of the destination on the front and the sender's address on the back. The process that encapsulates the data with addresses is called 'Layer 2' or the 'Data Link' Layer. Thus the format of the frame is:

SOURCE ADDRESS	DESTINATION ADDRESS	DATA

As the frame is passed around the ring, each computer compares the destination address to its own address. If there's a match, the computer reads the frame and has received the data. Computers that don't match just forward the frame on.

Ring-type networks are used today in Synchronous Optical Networking (SONET) and Synchronous Digital Hierarchy (SDH) networks.

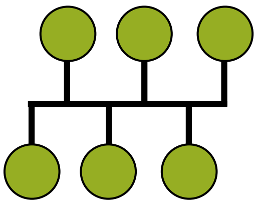
RESOURCES

DESCRIPTION	VIDEO	SCRATCH ANIMATION
Single ASCII character being transmitted	tinyurl.com/ycxoz3dy	helloworld.cc/2rinxzY
Word being transmitted using ASCII codes	tinyurl.com/y83vyauk	helloworld.cc/2JPZ6l2
Full meshed networked computers	tinyurl.com/y9sw68c8	helloworld.cc/2lbSHDC
Ring-based network	tinyurl.com/yb3yykhz	helloworld.cc/2rdYrll
Ethernet bus-based network	tinyurl.com/ybse8hg9	helloworld.cc/2HMHp9y
Ethernet hub-based network	tinyurl.com/y7zc2zzs	helloworld.cc/2jr5Uu0
Ethernet switch-based network	tinyurl.com/yas3z5sb	helloworld.cc/2HSNq0s
NIC sending data frame	tinyurl.com/y7sz9zj4	helloworld.cc/2KtpHpf
NIC receiving data frame	tinyurl.com/y7bz9xdd	helloworld.cc/2KzQksC

Bus

Another option to connect multiple computers together is to connect them all to a common wire. Initially, this was a thick coaxial cable, similar to a TV cable. Each computer connected to the cable with a 'tap', which was a spike in a clamp, tightened up with a nut. This visualisation of many computers being connected to a common wire coined the idea of similarity with a bus where people could get on and off as they wished.

The single wire meant only one computer could send at any one time, and the data would go to every computer on the wire. The technology was called Ethernet and used a set of rules called CSMA/CD to manage more than one computer trying to send at the same time. Mechanical problems with the taps and dry joints led to these networks being unreliable and difficult to fix.



An improvement used a thinner coaxial cable and special connectors called BNC connectors to make the connections. This was called 'Thin Ethernet' and the original cabling was retrospectively renamed 'Thick Ethernet'.

Just like a ring network, all the computers need an address. This is the Media Access Control (MAC), Ethernet, physical or hardware address. Data is encapsulated with a header containing the source and destination address to make a data frame. In early Ethernet networks, frames were received by all computers and each compared the destination address to its own address. If there was a match, the computer reads the frame and had received the data, otherwise the frame was just ignored.

Thus the format of the frame is:

SOURCE ADDRESS (6 bytes)	DESTINATION ADDRESS (6 bytes)	DATA (up to 1500 bytes)

Cabling

With Thin and Thick Ethernet networks, the electrical signals that carried the binary data were protected from interference by the braiding in the coaxial cable. This literally shielded the inner conductor by providing an electrical 'Faraday' cage around the core.

Improvements in technology resulted in being able to send the data over a pair of wires twisted together in the same way wires are twisted in a telephone cable. One pair is used to transmit data and the other pair to receive.

The cable is called Unshielded Twisted Pair (UTP) and is commonly, although incorrectly, known as Ethernet cable. Connection is made via RJ45 plugs.

Hub

To provide resilience and simplify connections, the bus was collapsed into a box called a hub. Each device connected directly to the hub on its own RJ45 port. Inside the hub, signals are received on one pair, are then regenerated, and just like the bus, transmitted out of all ports. This simple, reliable and cheap way to interconnect computers led to a high growth in the number of Local Area Networks (LANs) with multiple computers connected using an Ethernet hub. We call hubs 'Layer-1' or 'Physical Layer' devices because they just regenerate the electrical signals with no notion of the structure of the frame.

Switch

Hubs just forward data frames out of all ports because they have no knowledge of which computers are connected to which ports and have no understanding of the data they're forwarding. However, advances in electronics have allowed us to improve the efficiency of our Ethernet networks by putting some 'intelligence' in the hub. They can now inspect the frame and examine the source and destination Ethernet addresses. Clearly, the device is now much more than our humble hub and is called a 'switch'.

Initially the switch will not know the addresses of the connected computers so it defaults to a hub behaviour and switches incoming frames out of all ports. However, it learns which addresses are connected to which ports by examining source addresses on incoming frames which are stored in a table within the switch. Hence, future frames are switched to the right ports. We call switches 'Layer-Two' devices because they understand the headers at Layer Two, the Data-Link Layer.

The function of encapsulation is provided by the Network Interface Card (NIC) in the computer. Different interfaces such as wired, wireless, 3G/4G will all have different NICs. No matter what the media (except fibre), there's still the possibility of some electrical interference with the signal and spikes in the voltage. These 'spikes' result in a binary 0 being interpreted as a binary 1 or visa versa. It may not be obvious that an error has occurred, so we use a 'check-field' at the end of the frame to enable us to detect errors.

Thus our Ethernet frame is now:

SOURCE ADDRESS (6 bytes)	DESTINATION ADDRESS (6 bytes)	DATA (up to 1500 bytes)	CHECK-FIELD

When receiving the frame, we check the 'check-field' to see if any errors have occurred. If an error occurs, we discard the frame.

The next evolution was the interconnection of all these LANs and the birth of the Internet Protocol (IP). (HW)

SUBSCRIBE

Sign up today for a year - prices start at FREE!

THE MAGAZINE FOR COMPUTING & DIGITAL MAKING EDUCATORS
(Hello World)
Issue 5 | Summer Term 2018 | helloworld.cc
DATABASES AND BIG DATA
Why do many struggle with databases - and what can you do about it?
INSPIRING ENGINEERS!
Ideas and inspiration to get your students thinking like engineers
Coding and INCLUSION
Helping visually impaired students realise their potential
ORGANISING A COMPUTING DAY OUT
Where to go, and what to do when you get there...
PSSST! IS IT OKAY TO COPY?
Why building on existing code is great for teaching
PLUS FLOW CONTROL IN PRIMARY PROGRAMMING • HOW WE BUILT THE RASPBERRY PI • WHY IT'S WORTH TRYING GP! AFTER SCRATCH • MICRO:BIT • TANGIBLE PROGRAMMING IDEAS • 2018: YEAR OF ENGINEERING

SUBSCRIBE TODAY

- Get three term-time issues
- Have them delivered directly to your door
- Hello World is not available in stores!

(Hello World)

FREE
IN PRINT
For UK-based educators!

TO SUBSCRIBE VISIT:
helloworld.cc/sub1year

Not a UK-based educator?
Buy any issue for £6
Visit: www.store.rpipress.cc

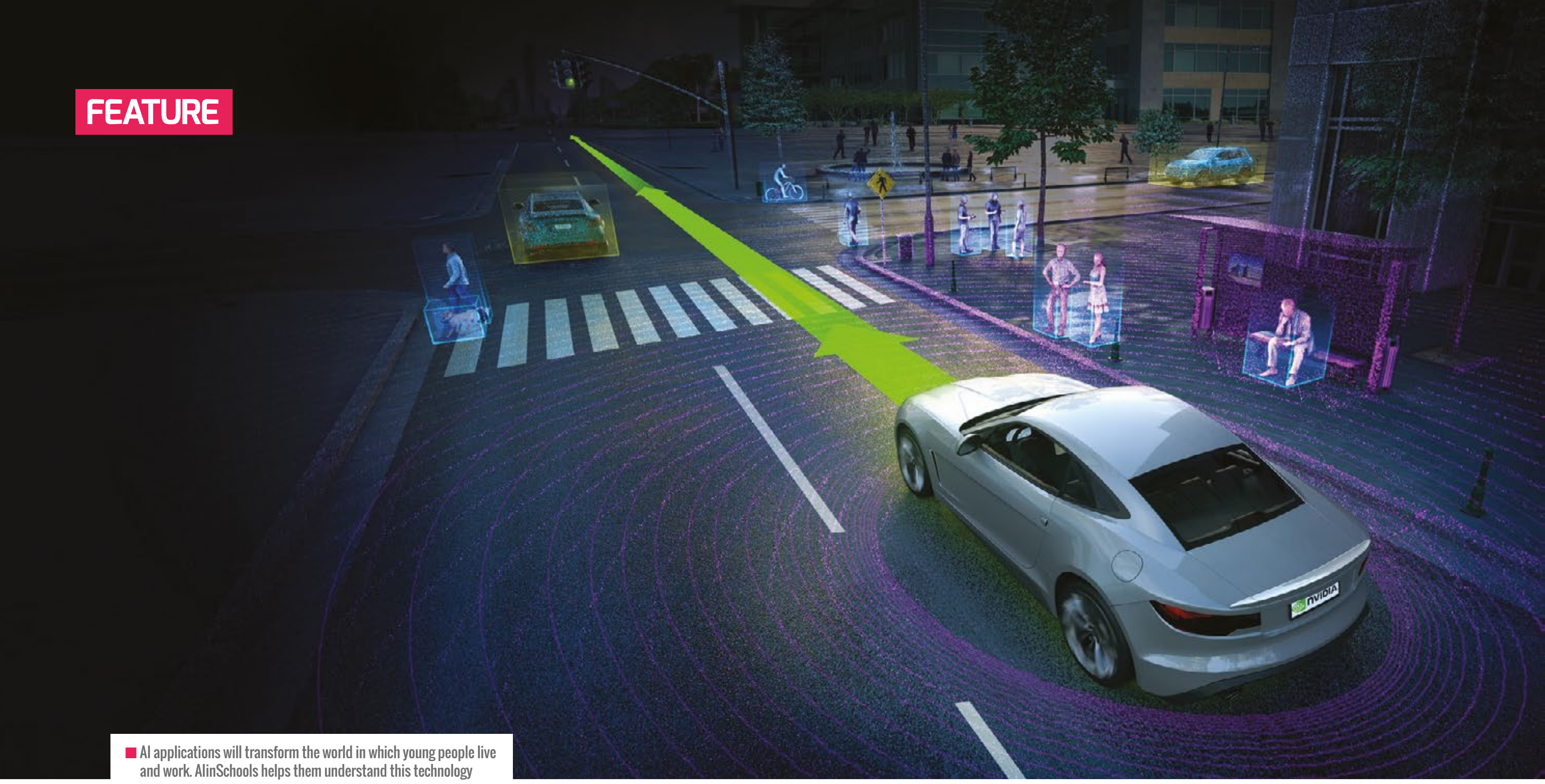
SUBSCRIBE ON
APP STORES

- Direct to your mobile
- For both Android & iPhone
- Back issues available

Available on the App Store

GET IT ON Google Play

FEATURE


■ AI applications will transform the world in which young people live and work. AIinSchools helps them understand this technology

AIINSCHOOLS PROGRAMME BRINGS AI TO THE CLASSROOM

AI will transform the lives of today's pupils. The AIinSchools programme provides a practical understanding of this technology to help them navigate the AI future

STORY BY Beverly Clarke

Artificial intelligence (AI) is rarely out of the news these days. Although it may seem like something from a sci-fi movie rather than part of our daily lives, most of us in fact are regular AI users. Each time we use a digital assistant like Siri, Alexa or Cortana, get a recommendation for a TV show from Netflix or have Google autocomplete our search term, we're using AI.

And these kinds of applications are just the beginning. The World Economic Forum estimates that, by 2025, 90 percent of jobs will require digital skills, and that 65 percent of children entering primary school today will work in jobs that don't currently exist. The lives of today's young people will be transformed in many ways by this technology, so it's imperative they receive a grounding in its origins and principles.

AIinSchools was created to help demystify and democratise AI. By putting AI into the hands of pupils, the scheme of work (SOW) aims to remove fear, spark inspiration and prepare a new generation for the intelligent industrial revolution. As well as equipping individuals to succeed in their future lives and careers, initiatives like AIinSchools have a vital role to play in ensuring the UK is positioned to reap the economic and social benefits of AI.

By providing a detailed SOW for teachers, AIinSchools aims to supply teachers with everything they need to explore and apply AI with their students. The teaching kit has been developed in accordance with the requirements of Key Stage 3 of the National Curriculum in England. It includes lesson plans, worksheets and activities, enabling teachers to deliver six one-hour lessons.

Understanding AI terminology and applications

The ability to think about and discuss AI critically relies on understanding key vocabulary. To this end, the SOW starts

EMBEDDING AI WITHIN THE SECONDARY CURRICULUM

The AIinSchools scheme of work covers half a term (six weeks) based upon one-hour lessons. Each lesson has ideas for starter and plenary activities, main, extension, stretch/challenge and support activities. A bank of questions is supplied to facilitate conversation around morals and ethics of artificial intelligence. Teacher presentation slides along with worksheets are also available. All resources are free. For further information, email **AIinSchools@nvidia.com**.

by explaining core terms like artificial intelligence, machine learning and deep learning, and makes clear how these often-confused terms relate to one another. Pupils also gain exposure to concepts such as 'broad AI', 'narrow AI' and 'super AI'. In addition, there's focus on concepts such as 'the Turing test' to give pupils an understanding of the roots of machine learning.

This is followed by a focus on deep neural networks, including where and how they're used. Understanding of neural networks is underpinned with a whole-class unplugged activity. Pupils further develop knowledge on the layers within a neural network and gain an understanding of how mathematics is used to adjust the weightings in a neural network. Throughout, the SOW provides prompts for teachers to question the class, so pupils gain a deeper understanding of concepts and can discuss them with peers.

While a basic understanding of terms is useful, it's equally important to inspire pupils

Unplugged activities

AI theory is supplemented by hands-on assignments. Pupils work on an image recognition task and gain knowledge of common AI frameworks such as Google Tensor Flow, Torch, MX-Net, MSFT and Caffe. Pupils gain knowledge of labelled datasets, supervised and unsupervised learning.

They also gain an appreciation of the contribution made to AI by different compute models as they utilise cloud-based graphics processing units (GPUs) to recognise their images with varying degrees of accuracy. There's no programming in this task or in the SOW, as not every pupil will be a programmer. The lesson shows how easy it is for anyone to build AI systems.

In all lessons, pupils are encouraged to look beyond the lesson and see real-world implementations of AI and to think of possible AI innovations. There's further development on how AI is changing our future with focus on agriculture,

"BY PUTTING AI INTO THE HANDS OF PUPILS, THE SCHEME OF WORK AIMS TO REMOVE FEAR AND SPARK INSPIRATION

with real-world examples of how AI is being applied in ways that are relevant and accessible to them. The theoretical elements of the SOW are grounded by a focus on AI usage in specific industries such as internet services, education, transportation, medicine and healthcare, media and entertainment, and helping the elderly/disabled.

Through discussion, these verticals combine to prompt an investigation of the concept of smart cities, including emphasis on big data generated in a smart city and how this is used, and an understanding of the Internet of Things (IoT). Pupils will then design their own smart city and look at a variety of considerations for a smart city. They can also use the knowledge and inspiration gained during this lesson to enter a national competition (currently in the pilot stage and run by the All Parties Parliamentary Group for AI) to design their smart city or workplace of the future.

shopping (automated services), cooking, transport including driverless cars (roads), medical diagnosis and assembly line manufacturing.

The concept of a chatbot is also explored. Pupils interact with a variety of chatbots and look at the way in which they work. Other emerging AI concepts such as affective computing are also delved into.

Cross-curricular links

Each lesson has a bank of keywords to facilitate discussion. Cross-curricular links are also provided with areas such as biology (neurons), mathematics (through understanding a weights matrix and also reading of charts when training the AI system), geography and eSafety.

Formative and summative assessment ideas and resources are given for each of the six lessons, and the AlinSchools scheme of work is pulled together with a short summative assessment, which ranges from multiple choice, to short answer to long answer questions. There's also an activity for pupils to look at famous women in AI. Lesson activities allow for paired, group and individual work.

Links with the national curriculum are made, and spiritual, moral, social, cultural (SMSC) development is promoted alongside British values. There are no specialist tools required to teach this unit, and delivery is suited to a specialist or non-specialist teacher.

AlinSchools was unveiled at the Bett education technology show in London on 26 January. Six schools will participate in an initial pilot project, following which the kit will be made available to secondary schools across the UK.

If you're interested in participating or would like to learn more, please contact **AlinSchools@nvidia.com**. (HW)

Beverly is a former secondary school teacher and Director of Computing who now combines roles as an author and education consultant for BCS Chartered Institute for IT. She is also a subject matter expert and Computing At School (CAS) Outreach Support for the South West region, a CAS Master Teacher and former hub leader. Her book, "Computer Science Teacher – insight into the computing classroom," was published in August 2017. She is also a CAS Board member As well as being a qualified teacher, she is a Chartered IT Professional (CITP) and National Professional Qualification in Senior Leadership (NPQSL) qualified. She is the subject matter expert for BBC Bitesize videos on Computational Thinking, and specialises in the Key Stage 3 element of the new Computing curriculum. Within schools, Beverly has successfully led departments, been seconded to "Sharing of Best Practice teams," and managed whole school projects. She also works with BCS Chartered Institute for IT, writing subject material and delivering workshops to trainee teachers. She has spoken at a variety of conferences and events such as the BCS Inclusion in IT launch, CAS South West regional conference, Future Sync conference, numerous CAS hub meetings and Surrey Subject leader meetings. Beverly is the creator of the AlinSchools scheme of work and resources.

Image courtesy of nvidia.co.uk

THE BEE-BOT
JUST WHAT IS A COMPUTER?

Could you explain it? Could the children in your class? We spend so much time learning how to use, program and control computers, we often don't take the time to recognise what they are

STORY BY Sway Grantham

Whenever I teach anything, I try to give the children a context – why is this something useful to know? I'm sure this is the same for many of you, as it gets children excited about the real-world impact they can have with this new knowledge. Computing is ideal for this. There's not much of our children's lives that isn't impacted by programmed technologies, from household appliances, to public safety measures, to everyday objects in supermarkets and shops, yet how many of these things do the children recognise as needing to be programmed or even having computers inside of them? Barely any.

What is a computer?

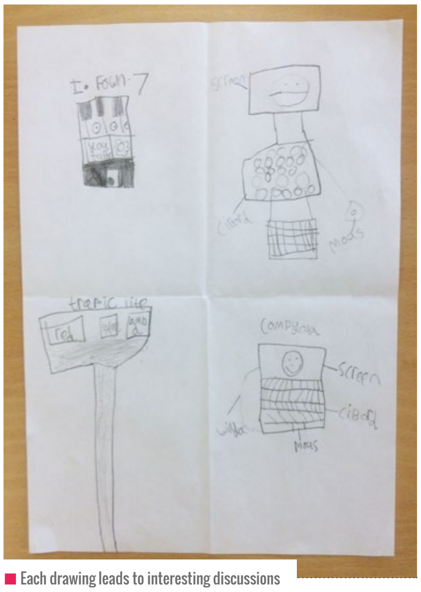
At the start of last year, I asked the children in Key Stage 2 the question: what is a computer? Their answers varied but many had similar themes. Here are my favourites:

- A piece of technology
- A keyboard and a screen
- A search engine
- A machine used for work
- A metal brain
- A machine with a keyboard
- Information device
- It's electric.

This simple activity can show us the beginnings of a wealth of misconceptions. For example, many children identified a computer as needing 'a keyboard' to make it a computer. There's also a belief that machine, technology, electrical device and computer are synonyms with one another. The other commonality is describing the computer's function, as if to define it we just need to know what it does. This definition greatly reduces the potential a computer has to only 'searching' or 'gaining information', and this understanding is going to hinder the children's acknowledgement of the application of computing to anything beyond personal use.

With younger children, first we get a piece of paper and ask them to fold it into quarters. In the first box, they have two minutes to draw a picture of a computer. Nearly all of them will draw a laptop. Then we discuss what they drew – did their laptops include a keyboard and a mouse? What about a screen? By acknowledging the parts of a computer, we can later explore which parts are necessary for a computer to work. Now we move onto box two. This time, I ask them to draw a 'different type' of computer, and I usually get a mixture of a desktop computer or a games console connected to a TV. Again, we talk about the parts. Is there a screen? What about a keyboard? Now we can have a discussion about there being no keyboard on a games console but instead there's a controller. I then repeat this process twice more, but this time I change the question to 'what objects do you think have a computer inside of them?'. Each drawing they do leads to interesting discussions, from traffic lights to remote control cars or ipads.

Is there a computer inside?


Each drawing leads to interesting discussions

Does it really matter?

I've already mentioned that I think it's important to show children a context for what they're learning, but recognising what a computer is and what devices have computers inside of them is more than that. After three years of a new Computing curriculum, my children can explain that an algorithm is 'a clear and detailed set of instructions that explain what you want the computer to do'. They can also tell you that a program is 'a set of instructions that runs on a computer to tell it what to do'. Both of these meet the National Curriculum requirements for Computing well. Following this, we might get out a programmable toy such as a Bee-Bot, and request they program it. However, to them, this isn't a computer. If it's not a computer, it can't run a program, so what are they learning from playing with it? Suddenly, their learning seems disjointed and nonsensical.

As the children get older, it becomes more and more important that they recognise the programs that are working all around them. The National Curriculum states that children in Key Stage 2 should 'work with variables and various forms of input and output'. This only makes sense if they can acknowledge what an input and an output are. Which parts of the computer are vital and which are optional? Inputs and outputs can vary – are you going to control your robot with a keyboard? Or do you want a voice-controlled robot? If you do, the input would be a microphone. Is your robot going to have a screen? It's unlikely that this is the most useful output for a robot. Earlier, those children who described a computer as having a keyboard are unlikely to be able to think of alternative forms of input that their program could make use of and thus not be able to achieve this part of the curriculum.

What should I do about it?

Even the youngest of learners at our school start learning about what a computer is and how to recognise one. They start with spotting buttons, wires, batteries etc and then we talk about what they do. If they recognise that when a button is pressed there are instructions to follow, they're beginning to understand what a computer is and where you're likely to find them. As children move through Key Stage 1, we begin the technology hunts, spotting buttons and discussing what might happen if we press them. This is where we start differentiating between those things that just use electricity and those that run a program. A light switch doesn't give an instruction to a light, it just breaks the circuit.

What happens when you press the button to cross?

“ EVEN THE YOUNGEST OF LEARNERS START LEARNING ABOUT WHAT A COMPUTER IS AND HOW TO RECOGNISE ONE

By Key Stage 2, we explore the world around us and try to work out what the algorithm would be. We use 'input' -> 'process' -> 'output' to decide if something has a computer inside or not. Each time we use this model, we reaffirm what an input and output are as well as the basic concept of programs running on computers. For example, what's the input on an iPad? How do we tell it what to do? There's a home button, a touchscreen, or we can talk to it using Siri (a microphone input). What code runs when we press the home button? Something like 'when button pressed, show home screen' and then the output? How do we know it's done what we wanted it to? We can see it on the touchscreen. This simple model allows us to test different machines or items of technology and tell if they're a computer or not.

One misconception I regularly hear is children referring to a monitor, or the interactive whiteboard, as a computer. Using this model, we can test this misconception – what's the input? There are some buttons. So what happens when we press them? It says 'no input'. What's the program it's running? It's not doing anything because there's no laptop plugged in. Then is it a computer? No. We now have a way to start conversations about whether or not a device is a computer and therefore whether or not a device is running a program.

As we work our way through upper Key Stage 2, we add to the challenge. What about when inputs are sensors? Can the children work out how automatic doors work? I've heard many amazing possible ways in which automatic doors can work, from a man watching a camera and pressing a button all day to heat detectors checking you're allowed to enter, but once again the children are exploring their misconceptions of the world and developing their understanding of how the subject of computing is applied all around us.

It's important that children learn what ubiquitous programming is so that they grow their understanding of a world they're a part of. Once you start these conversations, you never know where they'll end up! Take some time this week to ask your class: is a Bee-Bot a computer? (HW)

Sway teaches Computing at Giffard Park Primary School in Milton Keynes as well as being a digital technologies consultant. She was part of the first Picademy to become a Raspberry Pi Certified Educator and loves engaging in conversations about the pedagogy of computing. Find her on Twitter @SwayGrantham

GP, SNAP!, SCRATCH NEW STRATEGIES FOR NEW CONCEPTS

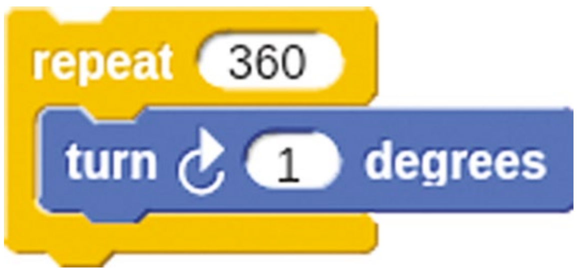
What's different with block-based programming?

STORY BY Sven Jatzlau and Ralf Romeike

Choosing the right programming language for your class is a tricky thing. The designers of the language have made choices about what it looks like and how easy it is to use. The choices made in block-based programming languages such as Scratch, Snap!, or GP make them very different from text-based languages like Java or Python. In these languages, we can find new concepts, and with them new approaches to solving problems. This also means we need to make learners aware of these new concepts and possibilities.

What's different?

In 2005, Scratch made its debut. Developed by the Lifelong Kindergarten Group at MIT, it represented a new approach to introducing programming to learners: based on the ideas of 'low floor, wide walls, high ceilings' by Seymour Papert, its popularity grew rapidly. At the time of writing, some 22 million projects, guides, animations, and games have been shared by a continuously growing user base of 19 million users. In schools, block-based languages display a similar success: compared to text programs, block-based environments enable students to achieve better results on average. They achieve these results faster, and with more motivation. At the same time, however, these block-based visual programs also introduced new concepts such as nesting of sprites, events, or first-class objects. While several of these concepts are found in Scratch, a number of concepts outlined in this article are only found in its sibling languages, Snap and GP. As they didn't undergo many of the simplifications found in Scratch, these languages lend themselves to secondary education. Teachers need to know something about the different concepts in order to more effectively teach them to their students!



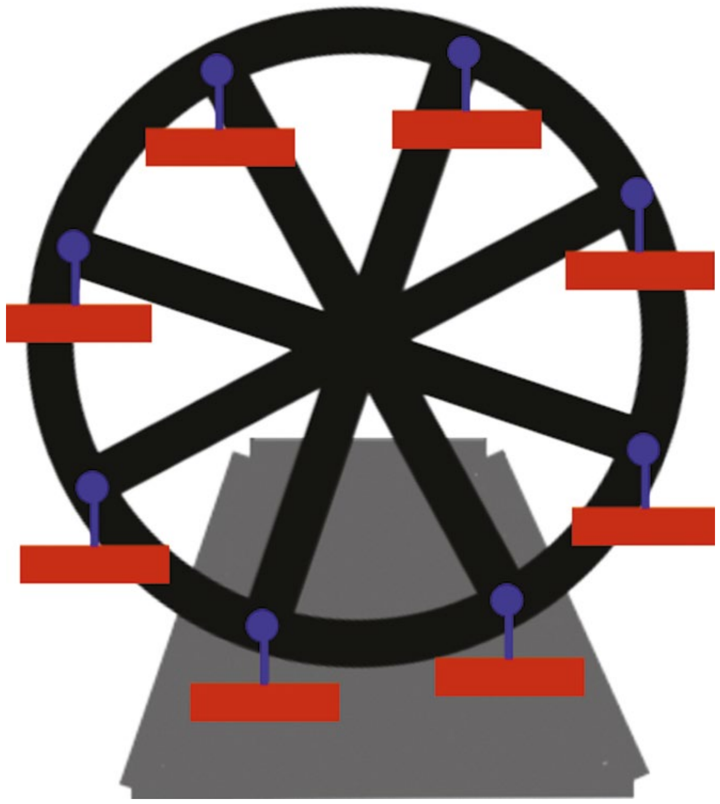
Instead of instantly executing the loop, making it hard to see what happened, the sprite will slowly rotate 360°

Delayed execution

Most of the new concepts exist for a pedagogical reason: making the language intuitive and easy to use. One of the ways this is reflected is that code scripts are executed slower than the environment (and the computer!) would normally allow. Without this delay, sprites could shoot out of the stage boundaries instantly, without the user being able to understand what happened, because they couldn't see. To protect users from experiencing this, all the loops, wait blocks and motion blocks were equipped with a delay. Snap! in particular takes this one step further: activating 'visible stepping' enables the user to not only see which block is currently being executed, but to control the execution speed manually. For debugging purposes, this is an invaluable tool!

Nesting

Visual programming creates new approaches to solve new problems. Sprite nesting is a great example of this. It creates a hierarchy between two objects/sprites by making one the owner (or 'anchor' in Snap!), and the other a part. While Scratch doesn't support the nesting of objects, Snap! and GP do. Dragging and dropping one sprite from the sprite palette onto the other one on the stage nests the two objects into each other. From that point onward, they'll move and turn as a single unit, reference each other using the anchor/ owner and parts blocks, but will still be considered two objects for the purpose of the program. This also means that they can still rotate themselves – it's easy to think of it as a part-whole structure, like a car: it consists of many single parts, like tyres, mirrors, the engine, etc. When the car drives and turns, all of its parts (naturally) turn and move with it. However, parts can be removed and added, and some can even turn by themselves (like the tyres) without it affecting the rotation of the entire car. In block-based programming, this concept enables the user to detect which part of an object collides with another object, to create composite objects consisting of smaller parts, and even to create physics-based simulations!


■ In this Snap! program, when the ferris wheel turns, the cabins turn with it. Each individual cabin constantly points downwards to simulate real physics

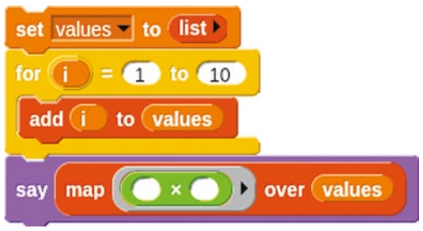
First-class objects

Whilst there are loads of complex definitions of the term, essentially the idea behind the concept is that it should be possible for every object in a programming environment to be used freely – there should be no limitation on the context in which an object can be placed: as the value

> BLOCK-BASED ENVIRONMENTS ENABLE STUDENTS TO ACHIEVE BETTER RESULTS ON AVERAGE

of a variable, an input to a procedure, etc. An example of this can be seen in Java: Integers, Strings and Booleans can be used as the input, or parameter, of a method, or as the return type. They can also be assigned as the value of a variable. It seems that there are no limitations on what we can do with these data types; this makes them first-class objects. But what about methods? Methods can't be assigned to a variable, used as the input to another method, or returned by another method. Do these limitations make them second-class objects?

In Snap! and GP, all objects are first class, therefore putting no limits on what the users can 'do' with them. This concept is closely related to the term 'higher-order functions': being able to store methods (or blocks in our case) in lists allows us to use them as inputs for other functions, therefore making them higher-order functions!


```
set values to list
for i = 1 to 10
    add i to values
say map ( ( ) × ( ) ) over values
```
■ Mapping the function over values returns a list of 1, 4, 9, and so on. Higher-order functions made simple!

Event-based programming

There have always been event-driven programming languages (events such as user input, sensor readings, or messages from other threads control the flow of the program). This is the type of programming that can be found in Scratch, Snap!, and GP. Because events are intuitively understood as something one can react to, learners instinctively program parallel solutions to problems: an object reacts to a button input with one script, and to touching another object with another script. If both events occur at the same time, the object does both things at the same time! Learners implement these solution without even realising why that's a hard thing to do in other languages. This opens up the possibility of learning about concurrency and dependency in simple, intuitive environments and contexts.

What does this mean?

The growing presence of these programming languages brings great, new possibilities to classrooms. At the same time, however, new teaching strategies are needed in order to teach these new concepts and their possible applications. The more concepts learners understand, the more options they have to choose from to implement their solution, and more complex problems may be solved. (HW)

WHAT TO DO AFTER SCRATCH? TRY GP!

GP lets students explore advanced computational ideas in a playful, blocks-based language

STORY BY John Maloney

GP, from the implementors of Scratch and Snap!, is a blocks-based programming system that supports general-purpose computation. Think of GP as 'Scratch meets Python'. Like Scratch, GP's built-in graphics and animation help students see and understand program operation whilst creating projects that connect to their own interests. Like Python, GP can do many things, such as process pixels, sound samples, data structures, music, statistical and scientific data, and much more.

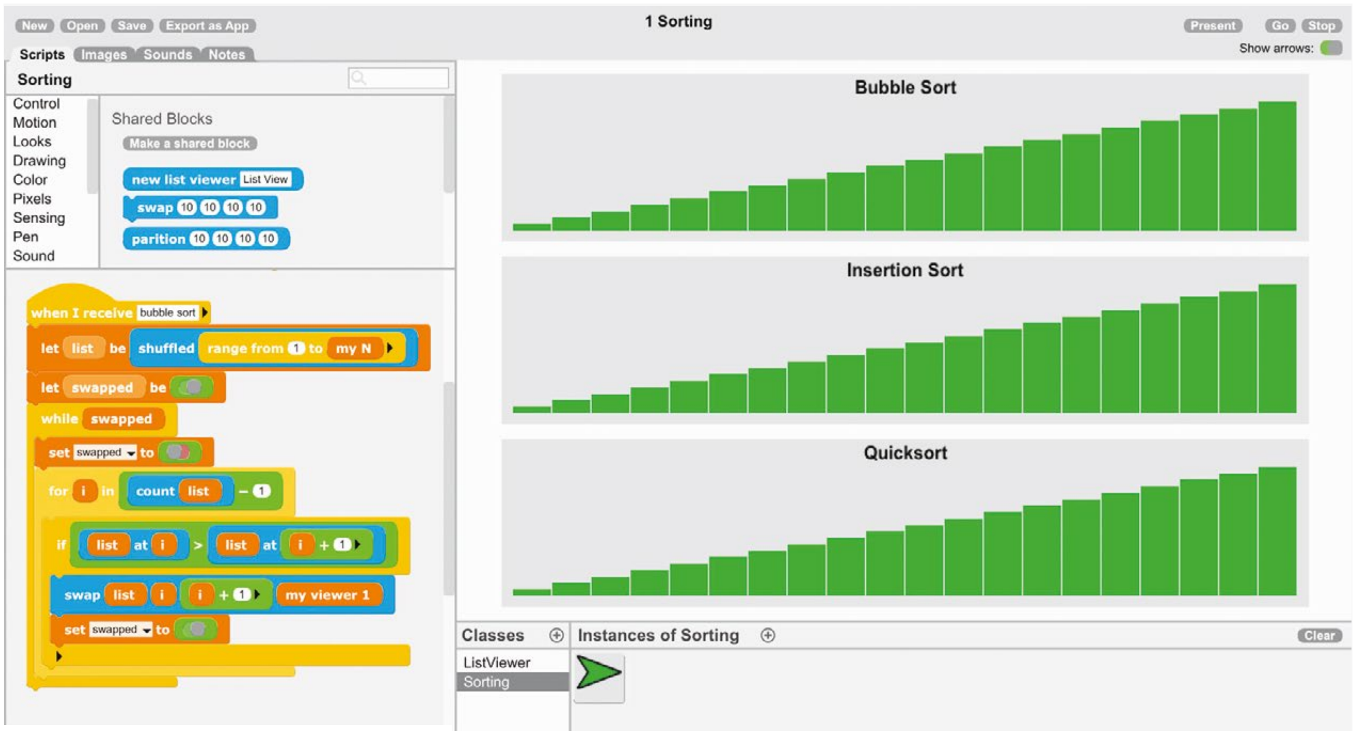
Students who start with Scratch often hit a wall when they attempt to jump into a text-based language like Python or Java. GP gives those students another path, allowing them to go beyond what's possible in Scratch without struggling with syntax.

Educators can use GP to explore Computer Science and STEAM subjects without having their class get bogged down by the steep learning curve of a text-based language. They can focus on concepts and ideas rather than programming mechanics, possibly engaging students more deeply as a result.

Who uses GP?

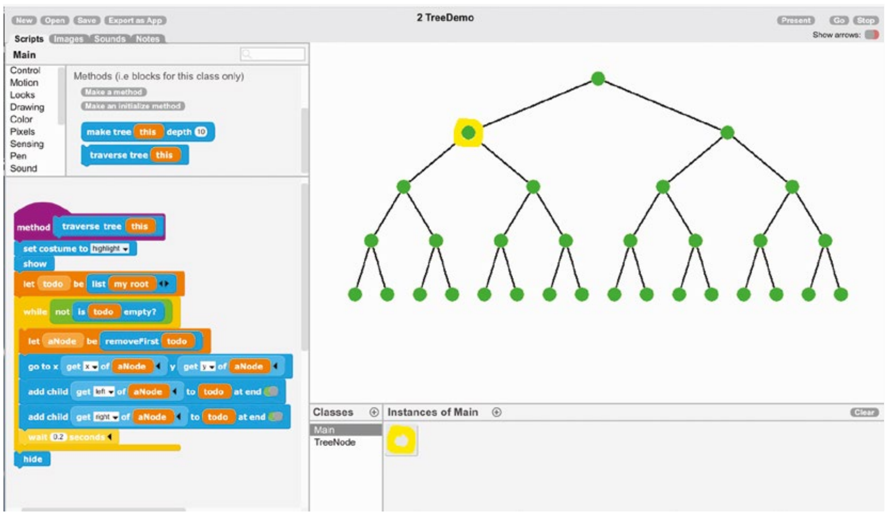
GP is being used around the world, with students ranging from Year 5 through to university. For example, GP is being used at a university in Germany to teach programming to science majors, with examples ranging from barcode scanners to celestial mechanics. This year, at a high school in Texas, GP was used in an Advanced Placement course to introduce class-based object-oriented programming before transitioning the students to Java. Finally, at the middle-school level, tens of thousands of students have done GP's popular 'Hour of Code' tutorials.

What can you do with GP? The gallery page on the GP website (**gpblocks.org**) shows some examples: manipulating the pixels of a photograph, simulating an ecosystem, exploring mathematical ideas



```
when I receive bubble sort
let list be shuffled range from 1 to my N
let swapped be
while swapped
  set swapped to
  for i in count list - 1
    if list at i > list at i + 1
      swap list i i + 1 my viewer 1
      set swapped to
```

Animation helps students see how Quicksort outpaces Bubble Sort as N increases. Students can implement and animate additional algorithms



EXTENDING GP

For advanced or curious learners, GP offers something that no other blocks-language offers: the ability to view the GP system itself, including its libraries and IDE, as blocks. Adventurous users can even extend the GP system.

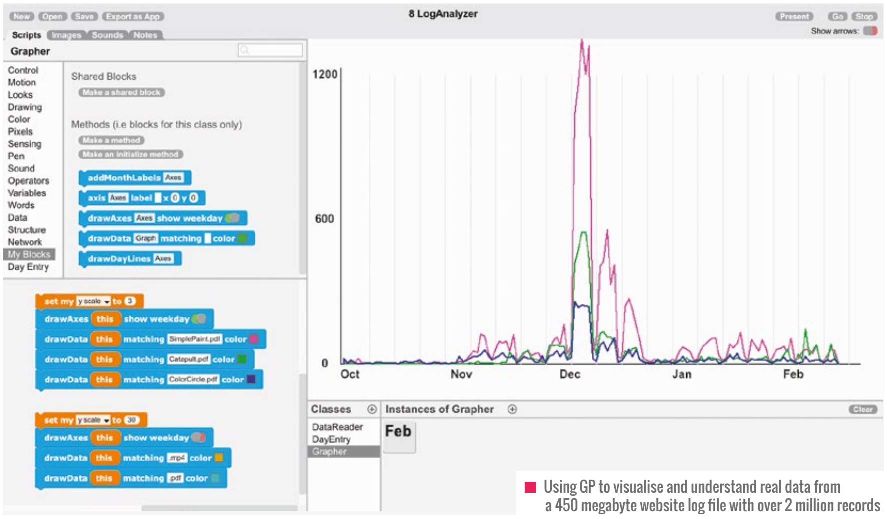
■ In a few hours, a teacher can create a tree visualisation framework to help students explore tree traversal and manipulation

such as the Mandelbrot set, or displaying a graphical view of a musical score as it plays.

Teaching CS with GP

For Computer Science teachers, GP can be a compelling way to teach object-oriented programming. The GP interface makes classes and instances visible and use conventional vocabulary.

In addition, since GP has Scratch-like animation and graphics built in, data structures can be made visible and algorithms can be animated with just a little extra code. For example, a teacher might create an animated version of



■ Using GP to visualise and understand real data from a 450 megabyte website log file with over 2 million records

> DATA STRUCTURES CAN BE MADE VISIBLE AND ALGORITHMS CAN BE ANIMATED WITH JUST A LITTLE EXTRA CODE

Bubble Sort, then challenge students to implement a faster sorting algorithm, such as Quicksort, or students might understand tree operations through animation.

Working with data

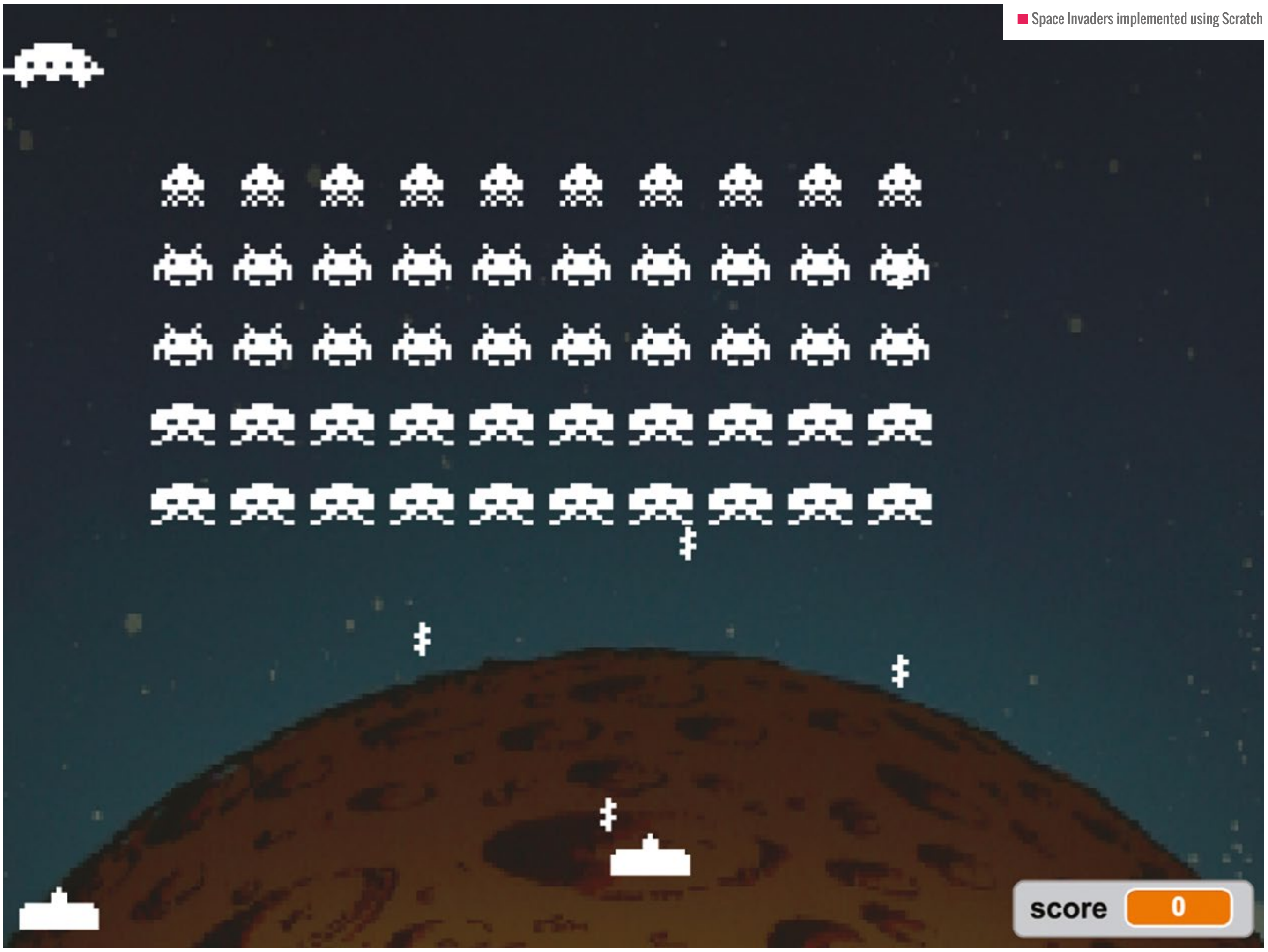
GP also allows students to work with real data such as text or CSV files. Students might be asked to write programs to count the number of unique words in text files of books from the Project Gutenberg website, then compare the working vocabularies of various authors (e.g. Jane Austin vs Louise May Alcott). Data analysis in GP can go beyond mere classroom exercises. For example, the author used GP to analyse and graph a 450 megabyte server log file containing over two million records to gain a better understanding of the GP website usage patterns over time.

Where to get GP

You can learn more about GP at **gpblocks.org**. GP runs in all modern browsers and on Chromebooks. It can also be downloaded as a standalone application that runs on Windows, MacOS, and Linux. The standalone version of GP is a bit faster than the browser version and can use local files, serial ports, server-side sockets, and other facilities that are blocked by the browser sandbox. It can also explore a GP project as a standalone executable, a feature that can be very motivating for students.

If you love Scratch and are looking for a way to take it to the next level, give GP a try! (HW)

John was the lead developer for Scratch for its first 11 years. For the past few years, he has been creating GP with Jens Mönig and Yoshiki Ohshima at several research labs. John has a Ph.D. in computer science from the University of Washington and has been working on live, beginner-friendly programming systems for over 25 years.



■ Space Invaders implemented using Scratch

COPYING IS ALLOWED!

Copying is important for learning, not just to teach skills but also for inspiration. Help give your class a flying start by encouraging it!

STORY BY John Arnold

Copying - everyone knows it's bad, right? Copying the schoolwork of the kid next to you or copying someone else's style in art? These are universally frowned upon. Shouldn't we develop our own style? Use our creativity and make something new? Well, ideally yes, but there's a lot to be gained from looking at the work of others, especially when you're developing a new skill like programming.

I teach Code Clubs in two local schools and I always begin the year by telling the kids my two golden rules. The first rule of Code Club – copying is... I pause to see if they'll fill in the gap... allowed! This usually raises a smile or a laugh. The second rule of Code Club – if you've made something cool, show it off! I have other rules – no playing games unless you wrote them yourself, no YouTube, don't talk about Code Club – but those first two are really important.

The first rule of Code Club

First of all, copying is a long tradition in programming. Professional programmers all build on the work of those who came before.

The use of library code and well-understood programming patterns is the bedrock of our profession. And I want the kids in my club to understand that there are no extra prizes for reinventing the wheel. If your friend on the next desk has written something cool and you want to do the same, steal their code. Give credit, sure. Modify it if you wish. But steal. Make progress. After all, is that so different from typing a game listing from a magazine into your Commodore 64 all those years ago? Of course not.

The real benefit of copying in this way is that those kids who really understand what's going on will pick apart what they've stolen and build on it. And those for whom the penny hasn't completely dropped yet will make progress and have fun. They'll produce more work and, hopefully, the more they do, the more they'll understand. And, let's face it, in an after-school club like Code Club, the point is first of all to have fun.

Job number one is to inspire!

So how about that second rule? If you've made something cool, show it off! Well this is about copying too. If a kid has done something they're proud of, they want to show it off. That's great because it gives you the chance to review what they've done, have them explain it to you and check their understanding. Review is a great teacher. But if they also show their classmates then it inspires the others. It gives them ideas. It gives them the chance to copy.

I see inspiration as my number one task in Code Club. I'm certainly there to teach them programming skills, answer questions and solve their problems. But, more importantly, I want them to see where programming can take them. I want them to have ideas and set about implementing those ideas. So when a child has made something cool, I want to celebrate that and share it around. I want others to be inspired by it and make their own versions of it.

The Scratch website understands this and provides a mechanism for kids to share their work so that others can build on it. When they look at a project they like, they can remix it. That's just a straightforward copy. The remixer gets all the code and all the images for free. They've essentially copied everything the first coder did and it gives them a flying start for their own project. I've found that different kids are interested in different aspects of software development. Some love the act of coding. Some love working on the graphics. And some are filled with ideas for cool game mechanics. We need all those skills in industry, and a remixed project gives each one what they need to explore their area of interest.

Stealing from the past

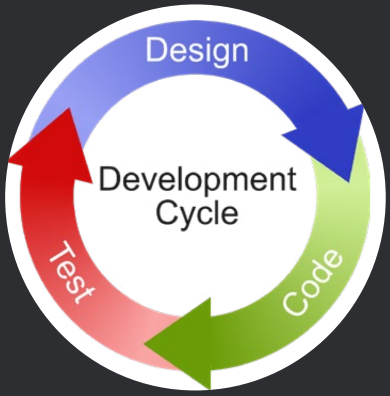
Inspiration is key in teaching kids programming. Once you're through the first few weeks of following the excellent Scratch projects on **codeclubprojects.org**, many will want to start making something of their own. They'll start out with an empty Scratch project and imagine they can create something simply by dragging random blocks around. Of course that doesn't work. Programming is all about telling the computer what to do, and you can't if you don't first know that yourself. So don't be afraid to steal designs too.

Help your kids generate game ideas by suggesting games they know from the apps world. Flappy Bird is one of my enduring favourites. Or classic arcade games often have simple rules that are easy to code. Work with them to describe in words what the rules of the game are. Then figure out what sprites you'll need and how each sprite behaves. Only when you've talked all that through and perhaps drawn a little on a whiteboard, let them start dragging Scratch blocks around. This is the design phase of the software development cycle and it's important because it's where they learn to pull apart a problem into smaller problems that can be solved in code.

With programming just as with art, we practise and develop our techniques and, eventually, when our skills mature and our tastes develop, our own style can emerge. Creativity doesn't always mean inventing something from a blank page. Give your kids starting points and inspiration. Encourage them to copy and to give credit. Very quickly you'll see them producing uniquely creative and unusual projects that you'll want to show off for them! (HW)

SOFTWARE DEVELOPMENT CYCLE

The software development cycle loops around. Begin with design, implement it in code and then test. Refine the design, write more code and test again. This iterative process helps refine the program until it's perfect!



John is a small business owner in Cheshire with a background as a professional programmer. He teaches Code Clubs in two local schools because he wants kids to love programming as much as he did when he learned on his beloved Commodore 64.

FEATURE

AGILE METHODOLOGIES IN THE CLASSROOM

Beyond giving students a taste of what working as a software developer entails, introducing agile practices into the classroom has lots of other benefits

STORY BY Harriet Ryder

Agile is an adjective applied to a range of methodologies that have been developed to improve the software development process and lifecycle. Being agile means moving and responding quickly. It means being able to respond to shifting requirements, industry changes and unforeseen problems. It usually involves cross-disciplinary teams of developers collaborating, planning and reviewing software together, and one of its key tenets is to work in short iterations, building the minimal product you need and then refactoring and improving it later.

It stands in contrast to the more traditional 'waterfall' approach, where a product flows down through several fixed stages – from initially deciding the requirements, through design, building, testing and then finally delivery. Between the first and final stages, years might have passed. Years is a long time in the tech world, and by that time the requirements might well have changed. But too late – the product's signed, sealed and delivered.

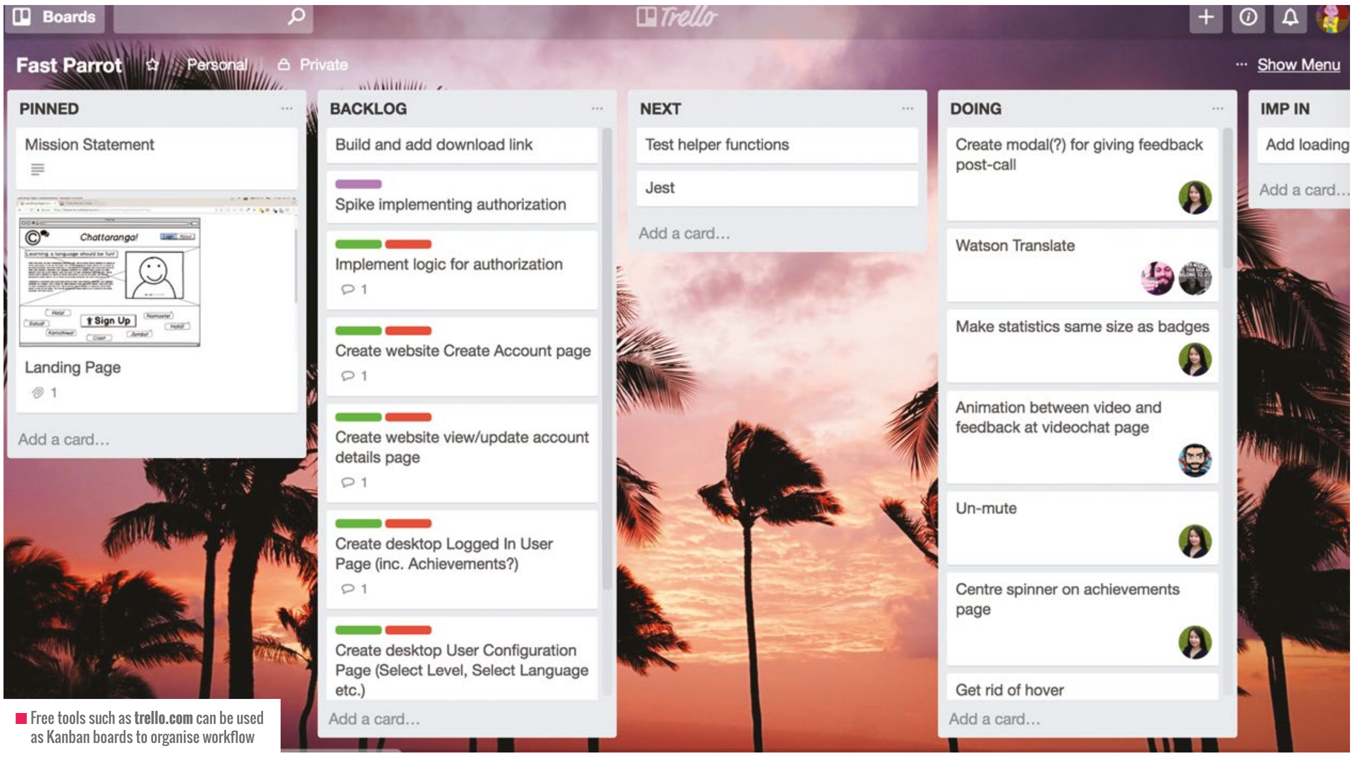
Why in the classroom?

I teach software development to adults, preparing them for work in the industry. Exposing them to common industry practices and structuring the classroom like a development shop is important in preparing them for work. However, the techniques and ideas that I'm going to talk about can be applied to all areas of life and provide tools for tackling any project or problem.

There are a whole range of methodologies and frameworks that can be used to help you work in a more agile way. I won't go into all of them, just a few favourites that we employ in the classroom every day.

Pair programming

For us, this is the most important agile methodology we use. During a project (which we call a 'sprint' and for us lasts two days), we pair students for about half the time. We give students a quarter of the allotted coding time initially to explore aspects of the problem alone, and then we randomly pair them up to continue working. When they pair, they have to start again, integrating ideas from both of their initial explorations. At the end, they split again and independently finish up the project in whatever way they prefer.



PINNED
- Mission Statement
- Landing Page

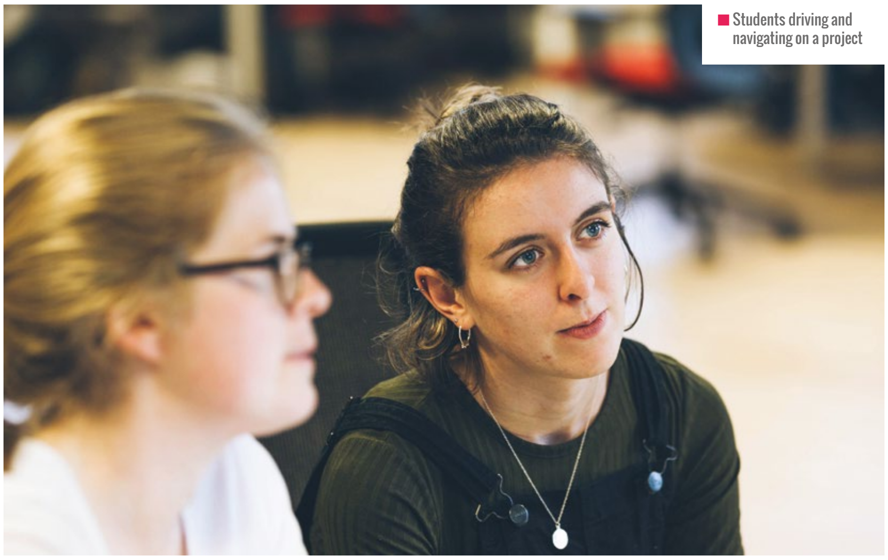
BACKLOG
- Build and add download link
- Spike implementing authorization
- Implement logic for authorization
- Create website Create Account page
- Create website view/update account details page
- Create desktop Logged In User Page (inc. Achievements?)
- Create desktop User Configuration Page (Select Level, Select Language etc.)

NEXT
- Test helper functions
- Jest

DOING
- Create modal(?) for giving feedback post-call
- Watson Translate
- Make statistics same size as badges
- Animation between video and feedback at videochat page
- Un-mute
- Centre spinner on achievements page
- Get rid of hover

IMP IN
- Add loading

Free tools such as **trello.com** can be used as Kanban boards to organise workflow


Students driving and navigating on a project

DRIVING AND NAVIGATING

The driver role:

- Is the one using the computer
- Listens to the navigator
- Implements the ideas of the navigator
- Can question the navigator

The navigator role:

- Directs the driver
- Explains their ideas
- Doesn't touch the computer
- Can use pen/paper to help explain their thinking

How to pair

Whilst pairing, students work on one single computer and follow a strict driving-navigating model, as explained in the boxout. Whilst pairing, students have no choice but to try to explain and communicate their ideas. This can be challenging at first, but continual practice at explaining technical concepts and ideas really helps students embed what they're learning and discover what areas they don't fully understand.

PRACTICE AT EXPLAINING TECHNICAL CONCEPTS AND IDEAS HELPS STUDENTS EMBED WHAT THEY'RE LEARNING

Every 30 minutes, we ask students to swap roles. This might involve using a version control system to push their work to a service like Github and then pull it down on the other student's computer, or it could just mean swapping who's using the same computer.

Scrum

Scrum is a framework for managing roles and workflow in an agile way. A few things we've borrowed from Scrum are the concepts of stand-up meetings and retrospectives. If students are working on a project over several days in a small group, we help them organise stand-up meetings at the beginning of the class, where they stand in a circle and each student has a few minutes to explain what they've been working on, what they found challenging and whether they've overcome it, and what they want to achieve today. We use this in combination with Kanban (explained below). We think it's really important for every student to hear what other students struggled with, as everyone will be struggling with something.

We also encourage group retrospectives, where upon finishing a sprint the group will discuss what went well, what didn't go well, and what could have been done differently. Sometimes, students will keep ongoing blogs or logs of their work on a project and use this during retrospectives to refer back to.

Kanban

The principle of Kanban methodology is to be able to visualise work and tasks to be completed. 'Kanban' means billboard in Japanese and that's usually what you use - some kind of billboard divided into columns. Tasks can be moved from column to column as they progress through the development workflow. Sometimes, the board is drawn up on a whiteboard, but free software such as Trello (**www.trello.com**) can also be used. You can also tag individuals on cards, add additional information and upload attachments.

Github also has a similar board system called Projects. Github Projects are great because you can integrate your cards with Github Issues and link to parts of your code.

Many students on our course end up using Kanban to organise everything – from independent studying, job hunting to general life tasks!

'Just make it work'

Another concept we spend a lot of time talking about is the Minimum Viable Product. Building an MVP means building the most barebones version of the product that will work. We discourage students from trying to create the most amazing, all bells-and-whistles product on the first attempt because who knows how the specification might end up changing later and how much time we will have wasted.

But beyond practical implication, stressing the importance of 'just make it work' encourages small wins, which boost confidence. Students can improve, refactor and rethink their initial attempt, but at least they've got something to show. Creating something that 'just works' also gives a starting point to talk about how we can make the code cleaner and more modular, or consider edge cases. (HW)

Harriet teaches software development to adults in one of the UK's leading bootcamps.

FEATURE

USING MAKERSPACES TO EXPLORE ENGINEERING IN THE CLASSROOM

Makerspaces are becoming a hub for STEAM in schools and they've become an invaluable tool in bringing engineering to the forefront of school programmes

STORY BY Nicholas Provenzano

Ella H helps repair the team robot after competition. Exploring robotics is a great way to get students involved in engineering

Teachers have begun to explore the power of bringing makerspaces to their schools and classrooms, and they're starting to see how they can fully support the various aspects of STEAM. While all aspects of STEAM are well supported in makerspaces, engineering has really taken a hold in many spaces around the world.

As a teacher, looking at new makerspaces and seeing all of the tools can be intimidating, but there's space for all levels of learning. As students and teachers explore engineering in a makerspace, their confidence with the various tools and concepts will grow, and the space will grow with them. It's best to break down the approach of utilising a makerspace to explore engineering into three levels: beginner, intermediate, and advanced. A student or teacher's level has nothing to do with age, but rather the skills they've acquired by utilising the space. To help you understand, I'll break down the levels and provide examples to explore in your space.

Beginner engineering level

We all have to start somewhere, and the beginner engineering level is where many people will begin their STEAM journey. At this level, students and teachers are going to spend their time exploring the design process. They'll identify problems in their home, school, or community and try to come up with solutions to those problems.

Designing solutions to problems is one of the main aspects of every type of engineer. It's a skill that can help anyone in any job or life situation. For the beginner level, students and teachers should use cardboard, popsicle sticks and other crafting supplies to explore what design and prototyping will look like.

Students explore the early stages of engineering by creating projects with littleBits in the school makerspace, The Knight's Forge

Starting with arts and crafts to prototype designs is a great entry-level tool to engage beginners in the engineering process

A fun project I've done with students is to give them a set number of large marshmallows, usually 10-15, and a handful of smaller marshmallows, and 30 pieces of long, uncooked spaghetti noodles. Once they receive these supplies, they're instructed to work in their group and build the tallest tower possible with the supplies provided. I give the students an hour, and we measure the towers at the end. I've done this with adults too and it's always fun. Students learn the value of taking their time, designing a possible solution, and then building. This is always more successful than building right away and figuring it out as they build. This entry-level project with students and teachers is a great way to start exploring what engineering looks like using basic tools and arts and crafts. Once everyone feels comfortable in this area, it's time to explore the next level.

Intermediate engineering level

For the intermediate level, it's time to explore the digital side of engineering in STEAM. Scratch is an excellent block-based coding system that's perfect to explore for people new to coding and design. Scratch can help any user new to coding begin to build a foundation in computer engineering that could grow into so many different areas of programming and design if they stick with it. A great example to weave this type of design and engineering into a curriculum that utilises the makerspace was a project created by one of my Maths teachers. She had students create their own arcade games using Scratch to help them gain a better understanding of how objects move along an X/Y plane.

The students had to design the game they wanted and explore how to make that game possible using the code available in Scratch. As an extension to this project, some students designed their own controllers using Makey Makey and cardboard. Students spent time designing their controller to be able to control the game they created. This is a wonderful example of students taking the design and engineering process from concept to product. As students begin to become more comfortable with the tools that are available in the makerspace, they'll push themselves creatively and start to focus on larger projects that require more complex engineering and design. The same is true for the teachers who are learning in the makerspace as well. Their projects will increase in complexity to support the students who are fine-tuning their skills. The advanced level for makerspace engineering is where students and teachers can really let loose.

Advanced engineering level

For the third, and final, level, students and teachers have the opportunity to dive into all of the different aspects of engineering that interests them. This is a great time to support students as they look to bring the digital and physical world of engineering together to create a multitude of projects. The sky's the limit for this skill level for students and teachers. Exploring robotics and IoT are common projects for those looking to stretch their engineering and design muscles.

A wonderful example of this is when I had a student use littleBits and their code kit to create her own robot. The robot had a motion sensor and speaker that would play sounds based on motion it detected. It could also be controlled through a bluetooth connection on her phone. This student is in 6th grade, and she was able to design, code and build her own robot. She gained the confidence to do this by exploring physical computing earlier in the year with Raspberry Pi and modelling ideas in cardboard. She took all of the skills she learned over the course of the year and put them together to create her own robot using littleBits. Her build is a testament to the power of STEAM in the classroom and how engineering can be a great way to introduce many important STEAM concepts over the course of a student's education.

There are many different parts to engineering, and a student or teacher might be advanced in one aspect, intermediate in one, and a beginner in another. It's important to remember that every person is on their own personal educational journey and they'll arrive at different times. By having options for these learners to explore at all levels, it encourages them to keep going and push themselves to get the most out of their learning.

As educators look to expand the types of lessons that harness the power of STEAM content, check out your local makerspace and see how the tools there can empower you and your students. (HW)

" A MAKERSPACE SHOULD HAVE ALL THE TOOLS STUDENTS NEED FOR THEIR EDUCATIONAL JOURNEY

TOOLS FOR EACH LEVEL

BEGINNER
Cardboard, scissors, glue, duct tape, string, yarn, popsicle sticks
INTERMEDIATE
Scratch, Raspberry Pi, Arduino, 3D printers, Makey Makey
ADVANCED
littleBits code kit, Python, Raspberry Pi Sense HAT, various sensors, motors, servos

Makerspaces are a great spot for young girls to explore the different areas of STEAM with their friends

In a makerspace, students have access to a variety of different tools to support them as they explore the different aspects of engineering. Some students will be focused on 3D prototypes, while others will dive into computer engineering and explore writing code. No matter what they choose and what level they're at, a makerspace should have all the tools students need for their educational journey.

Nicholas is a technology coordinator and makerspace director at University Liggett School in Michigan. He's also an author, speaker and consultant. He writes on his website, TheNerdyTeacher.com, Edutopia.org, as well as many other prominent educational websites. He has been awarded the Technology Teacher of the Year by MACUL and the International Society of Technology in Education. Nicholas is a Google Certified Innovator, ASCD Emerging Leader, Raspberry Pi Certified Educator, and a TEDEd Innovative Educator. His best-selling book *Your Starter Guide to Makerspaces* is available on Amazon. Nicholas shares plenty of nerdy things on Twitter and Instagram at @TheNerdyTeacher.

TEACHING DATABASES USING BIG DATA

Databases are fundamental to modern society. So why do many young people (and their teachers) find the topic less engaging than other parts of our courses?

STORY BY Tony Harkins

There has been major curricular change in Scottish computing education in recent years. Previously, there have been separate qualifications in Computing and Information Systems. To generalise, the former focused on programming and the second on databases. The new qualifications, introduced in 2014, combined these two separate qualifications into one, new qualification called Computing Science. A recent update has introduced SQL queries at all levels of Computing Science.

At a recent meeting with local computing teachers, we were discussing, first, the importance of databases in the study of computing science and, second, how it can be difficult to engage pupils in a topic that can be drier than programming or designing websites.

How can a topic that underpins so much of modern technology be seen as boring or irrelevant by pupils who, unwittingly or not, interact with huge datasets multiple times a day? In this article, I'd like to explore an approach to beginning to learn about databases that show pupils the relevance of understanding and being able to extract meaning from modern datasets.

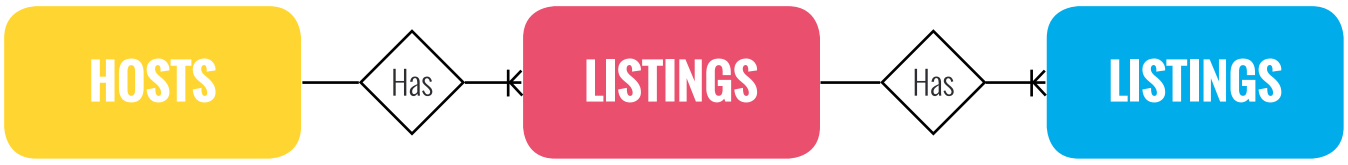
Big data

Imagine an introductory database lesson that talks about the millions and billions of records and fields contained in the databases of Amazon or Twitter. Pupils, keen to have an insight into the technology behind these huge enterprises, are then directed to Microsoft Access, where they have to create a single table, declare a few fields then create a handful of records in an Address Book or Music database. This discrepancy, between the reality of huge, enterprise-scale databases and what pupils can actually create, is huge. How many amongst us have had pupils enter five or 10 records into a database – busy work with no actual thought or educational value? Why do we need a database to store 10 records? A spreadsheet or even a table in Word would allow us to store and see the information at a glance.

To see the need, the power and the value of databases, we must use big data; datasets with thousands of records, data that requires queries and sorts in order to tell us something meaningful. Multiple, linked tables can then show why careful design of databases is so important.

Close to home, Glasgow City Council and the Scottish Government publish a range of datasets that could be used in class. **helloworld.cc/2HHHFCu** and **helloworld.cc/2w8RBDY** have datasets on crime, education, environment and population amongst others. We could answer the question 'Does it always rain in Glasgow?' definitively by analysing the data at **helloworld.cc/2IbUoAK.**

The Scottish Index of Multiple Deprivation, **helloworld.cc/2HKf0wB,** is



In a simplified AirBnB database, one host can have many listings, and one listing can have many reviews. Using Edinburgh data, over 6,200 listings from around 4,400 hosts had been reviewed over 100,000 times

Field(s)	listing_url, summary, price, bedrooms
Table(s)	Listings
Search criteria	Bedrooms > 1, price <=60, zipcode=EH99%
Sort order	review_scores_rating Desc

By identifying which tables are being queried, you can spot any joins that are required. Thinking at this stage about fields, criteria and ordering leads naturally to writing the query in SQL

IMDB

Many of our young people will be big fans of online streaming services, and will likely be familiar with the Internet Movie Database. Datasets from IMDb and its vast library of films and television series can be found at **datasets.imdbws.com**, with information about the data at **www.imdb.com/interfaces**. This could be an engaging context to explore and manipulate.

already an excellent, interactive example of data being used to analyse and compare huge amounts of socio-economic data. Simply comparing Glasgow and Edinburgh shows stark differences, but what can we say about the educational, health and employment opportunities for children growing up in rural versus urban areas? Drumchapel and Bearsden are neighbouring areas of Glasgow but have vastly different opportunities for young people growing up a few streets apart.

AirBnB

AirBnB has disrupted the world of travel and accommodation, and the site **helloworld.cc/2JFzaso** allows you to download vast amounts of data for many of the most popular destinations in the world that has been 'scraped' from the site. This context allows us to see the real-world website and the database behind it.

These datasets still require some analysis and work before they can be used in a classroom situation. Using the Edinburgh data, **insideairbnb.com/edinburgh**, I downloaded listings, hosts and review data for thousands of destinations in the capital and imported this into Access. Some fields required a little work to be usable. Creating a relational database of the data allows us to illustrate the cardinality of the data.

With the data, too much to take in by looking through the tables, we can start to construct queries. If I wanted to stay near the Scottish Government building in the EH99 postcode area, but could only afford £60 a night for at least two bedrooms, which table(s) would I look in and what criteria would I use?

The SQA exam board plan queries as shown above.

It's then trivial to code this query:

```
SELECT listing_url, summary,
price, bedrooms
FROM Listings
WHERE bedrooms>1 AND price 0
<=60 and zipcode LIKE 'EH99%'
ORDER BY review_scores_rating
DESC;
```

Looking at a single listing on the AirBnB website will show you the listing, some host information and a summary of their reviews. What better illustration of a join? This view is created dynamically using data from all three tables.

Data on InsideAirBnB is updated regularly. Pupils can click on the URLs contained in the database and verify that the data they're examining is for a real host with a real listing in a real city. I believe this authenticity and the quantity of data used will start to show pupils the value of database management software. The focus on SQL will mean they don't become experts in Access but, instead, understand the underlying structure of databases and the language used to query this data.

Then we can have other discussions too. Is it legal to 'scrape' data? None of the downloaded data is hidden or hacked, but do we have the right to extract, store and manipulate it? How can bot software be programmed to scrape the data? Why might the web server view the scraper software as some kind of attack?

Then we can look at the social and economic implications of AirBnB. Would you like to live alongside someone who lets out their apartment 250 nights of the year? What impact has the site had on the hotel industry? What impact does it have on cities that welcome lots of tourists each year? Does AirBnB drive up rents and restrict availability for local residents?

"TO SEE THE NEED, THE POWER AND VALUE OF DATABASES, WE MUST USE BIG DATA, DATASETS WITH THOUSANDS OF RECORDS

There are lots other sources of large datasets, so the challenge will be finding examples that will grab the attention of the young people in our classes, which they'll want to manipulate and manage. (HW)

Tony is head of Computing Science, St Aloysius' College, Glasgow.

AGE RANGE

7 - 10 years

LESSON TYPE

Physical Computing

REQUIREMENTS

- Micro:bit
- PC/laptop
- Internet access
- Torch

LIGHT UP YOUR WORLD

In this lesson, you'll create a light sensor using the micro:bit to indicate when you need more light for your classroom

STORY BY Fiona Hall

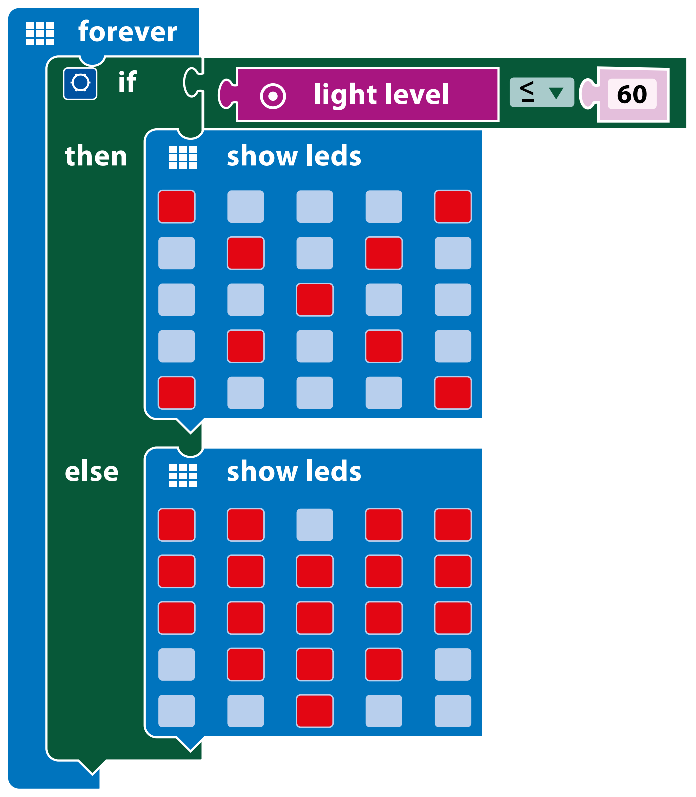
Enable your students to become inventors and engineers using the micro:bit and JavaScript Blocks Editor. This basic lesson gets children excited about what they can create with a few simple blocks of programming, a burgeoning imagination and a small but powerful device.

Children will use a few programming blocks to put together a program that will display an image when the light level in the classroom is too low. They'll also program it to display a different image when the light level is acceptable.

This lesson brings programming from the computer to a tangible device and enables children to explore physical computing in a supported environment.

Students will further their programming confidence and ability whilst controlling a physical system. They'll cultivate their imagination to consider how they could develop their sensor invention further – what could they create? How would they program it? Could it be used for good or for evil?

Allow children the time and flexibility to explore the



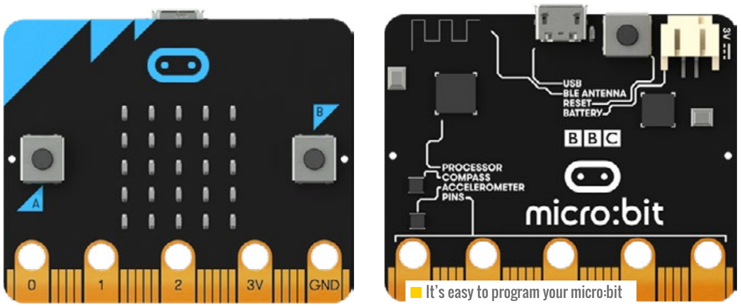
```
forever
  if light level ≤ 60
  then
    show leds
      🟥 ⬜ ⬜ ⬜ 🟥
      ⬜ 🟥 ⬜ 🟥 ⬜
      ⬜ ⬜ 🟥 ⬜ ⬜
      ⬜ 🟥 ⬜ 🟥 ⬜
      🟥 ⬜ ⬜ ⬜ 🟥
  else
    show leds
      🟥 🟥 ⬜ 🟥 🟥
      🟥 🟥 🟥 🟥 🟥
      🟥 🟥 🟥 🟥 🟥
      ⬜ 🟥 🟥 🟥 ⬜
      ⬜ ⬜ 🟥 ⬜ ⬜
```

Programming to complete light sensor task

THE CHALLENGE

- Use JavaScript Blocks Editor
- Include images indicating when more light is needed and when it's not
- The 'too dark' light level should be ≤60
- The program should run constantly

THIS LESSON BRINGS PROGRAMMING FROM THE COMPUTER TO A TANGIBLE DEVICE



It's easy to program your micro:bit

programming in their own style. They may use a longer or less efficient programming script, but they'll have developed their confidence and extended their knowledge en route.

Where time permits, extend the learning by allowing students to develop their sensor further – could they add a third light level? Could they add sounds that emit or play when the light hits a certain level? Could they develop an eco-sensor that detects when the lights are too bright or not needed? Give them the opportunity to develop their innovation and then to present their ideas to the class. (HW)

Fiona is a working mum, Brummie, ComputerXplorers owner who loves all things techy. Visit www,computerxplorers.co.uk for more about ComputerXplorers and details of free micro:bit workshops during British Science Week.

ALTERNATIVE ACTIVITY IDEAS

How might this work better for alternative age groups and lesson types? Explore alternate lesson plan types and give 10-15 words for each one in the following example format:

Explore and make

There are numerous ways you could allow learners to implement this physical computing session, on a variety of levels. Here are some ideas for modifying the lesson to suit a younger audience

5-6 years - Physical Computing

- Give children the micro:bit fully loaded with the programming for the lesson. Get them to experiment with the device to see the images change on the LED screen when they partially cover it with their hand
- Discuss when this might be used in real life
- Discuss how the device is controlled by programming. What is programming?

5-6 years - Unplugged

- Explore the basic algorithm for the light sensor (when dark, show image a; when light, show image b)
- Draw two images (one for light and one for dark)
- Simulate the algorithm using children, a torch and their newly created images

FURTHER READING

Full ComputerXplorers Lesson plan:

helloworld.cc/CXFreeDownload

More micro:bit ideas:

www.microbit.org/ideas

ASSESSMENT

- How did you ensure the program ran constantly?
- How did you test your program?
- Do you think your programming could be improved? How?
- How could you develop this program further?

AGE RANGE
8-11 years

LESSON TYPE
Computing/ Maths

REQUIREMENTS
Any device that will run Logo

LOGO TO ILLUSTRATE COMPUTATIONAL THINKING

In this lesson, we'll use Logo to create five characteristics of computational thinking

STORY BY Graham Hastings

For the non-specialist teacher and many pupils, computational thinking can feel quite a vague and abstract concept. The fact that there's no fixed definition of the term only serves to exacerbate this feeling.

This article illustrates five common characteristics of computational thinking using concrete examples created with the programming language Logo. Logo has an engaging graphical output that's produced as an animated cursor, usually called the 'turtle', moves about the screen, leaving a visible trail behind it. The version of Logo we use in my school is accessed via the free online Logo interpreter at **helloworld.cc/2HOsp6Q**. We've set up individual accounts for our pupils so they're able to save and retrieve their programs online.

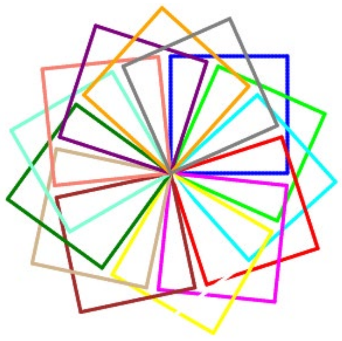
Five characteristics

Computational thinking isn't merely to do with computers, it's an approach to tackling problems that can be applied in any and every situation where a complex problem needs to be solved.

As computing activities frequently involve solving a problem of one kind or another, computational thinking is a vital competence for those engaged in the subject of computing.

The most common characteristics of computational thinking are:

Decomposition: taking a complex problem and breaking it down into a series of smaller, more manageable problems.
Pattern recognition: it helps to examine the small problems for similarities or 'patterns'. Recognising these patterns can help us to solve complex problems more efficiently.
Abstraction: focusing only on the important details, while ignoring irrelevant information.
Algorithms: simple steps or rules in a sequence that will solve each of the smaller problems and therefore the problem as a whole.
Generalisation: applying a problem-solving process to a variety of similar or related problems.


Flower motif illustration of the five characteristics

The illustration

I've used the flower motif above to illustrate the five characteristics – see steps 1 to 5 below.

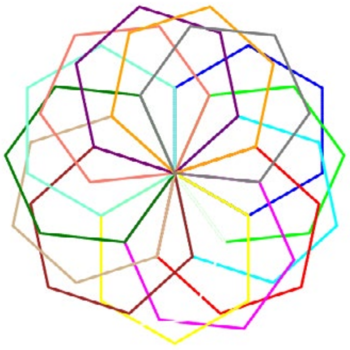
Step 1 The flower motif is complex but the image can be broken down, decomposed, into a number of recognisable parts. Each petal is a square; the petals are different colours and they're arranged around a central point; the angle between each petal is the same.

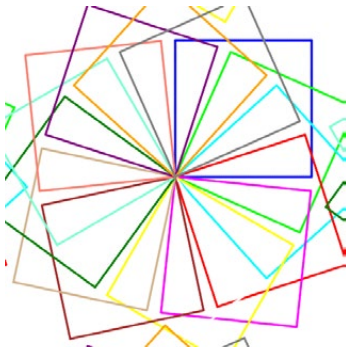
Step 2 Looking closely at the flower, it's possible to recognise a number of patterns in its construction. The centre of the flower is at the centre of the graphics window at position 0, 0. The flower consists of a repeating pattern of squares. Each square is drawn over the previous square, following a clockwise rotation by 24° and a colour change. The side of each square is 100 Logo units long.

Step 3 The important details we need to focus on are the lines that make up the sides of each square and the angles at their corners;

WHY SHOULD WE TEACH LOGO?

The Computing POS specifies that children should write programs in more than one language. Logo is an excellent alternative to Scratch as it's closer to a textural language yet uses very simple syntax. Logo is an engaging language that children thoroughly enjoy working in.


Replace the square with a hexagon


Double the length of the sides of the squares

the rotation of each square to the right by 24°; and the colour change. The removal of irrelevant detail when analysing a problem is known as abstraction. We don't, for example, need to know that there are 15 petals or that triangular shapes can be seen radiating from the centre of the flower. We don't even need to worry about the actual colour of each petal, the size of the square or the thickness of the lines.

Step 4 If we create a plan to solve each of these small problems, we can put all of the plans together to produce an algorithm that will solve the problem in its entirety. This is an algorithm.

Move the turtle to the centre of the graphics window. Repeat the following 15 times:

- Repeat four times, draw a line then make a 90° turn
- Rotate 24° to the right
- Add 1 to the colour value to change the colour.

Step 5 We can adapt the algorithm we've produced to draw a range of different flower patterns. This is known as generalisation. For example, we could replace the square with a hexagon, change the angle of rotation for the petals from 24° to 36° or double the length of the sides of the squares. The results of these generalisations are illustrated above and below.

Turning the algorithm into code

Once we have an algorithm, we can create the code to control the Logo turtle that will draw the flower. To make it easier to understand, debug and generalise, the code has been written as a series of procedures that are 'called' when the main program procedure, flower, is run. These are the procedures used:

position moves the turtle to the centre of the graphics window.

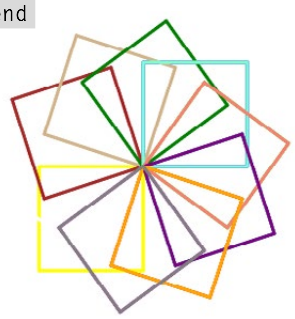
```
to position
setxy 0 0
end
```

side draws a line 100 pixels long.

```
   to side
fd 100
end
```

turn turns the turtle 90° to the right.

```
to turn
rt 90
end
```


Increase the angle of rotation for the petals from 24° to 36°

BACKGROUND INFORMATION

For some background information on Logo and its history, visit the Logo Foundation website:

el.media.mit.edu/logo-foundation

square calls the side and turn procedures four times to draw a square.

```
to square
repeat 4 [side turn]
end
```

flower is the main program, which begins by calling **position**.

In the next line of code, for **[i 1 15 1]** is a neat construct which does two jobs. It has to increment (add 1 to) the variable i which we're using to count the number of squares we've drawn and give a new input value (known as an argument) to the setpencolor command.

```
   to flower
position
for [i 1 15 1]  [setpencolor
:i    square    rt 24]
end
```

Logo colour codes:

```
1: "blue", 2: "lime",
3: "cyan", 4: "red", 5:
"magenta", 6: "yellow", 7:
"white", 8: "brown", 9:
"tan", 10: "green", 11:
"aquamarine", 12: "salmon",
13: "purple", 14: "orange",
15: "grey".
```

HOW DO I MAKE THIS HAPPEN?

Find a free version of Logo that will run on the devices that you use with your pupils. In my school, we prefer to use online applications when they're available, and for Logo we use **logointerpreter.com.** There are versions of Logo for the Mac and Windows operating systems. Snap!, a Scratch-style block editor, is also available from **snap.berkeley.edu.**

PRIMARY PROGRAMMING DESIGN

What's design in primary programming? What does it look like? How can we help pupils develop all the skills needed to develop design agency?

STORY BY Phil Bagge

If we want pupils to develop agency in programming, including them in the design process is important. Before we can do this, it helps to think through what type of design elements pupils may need. The problem is design means lots of different things to lots of different people. In this article, I suggest a useful framework for discussing and planning primary programming projects before suggesting ways we might use it.

Algorithmic, appearance and structural design

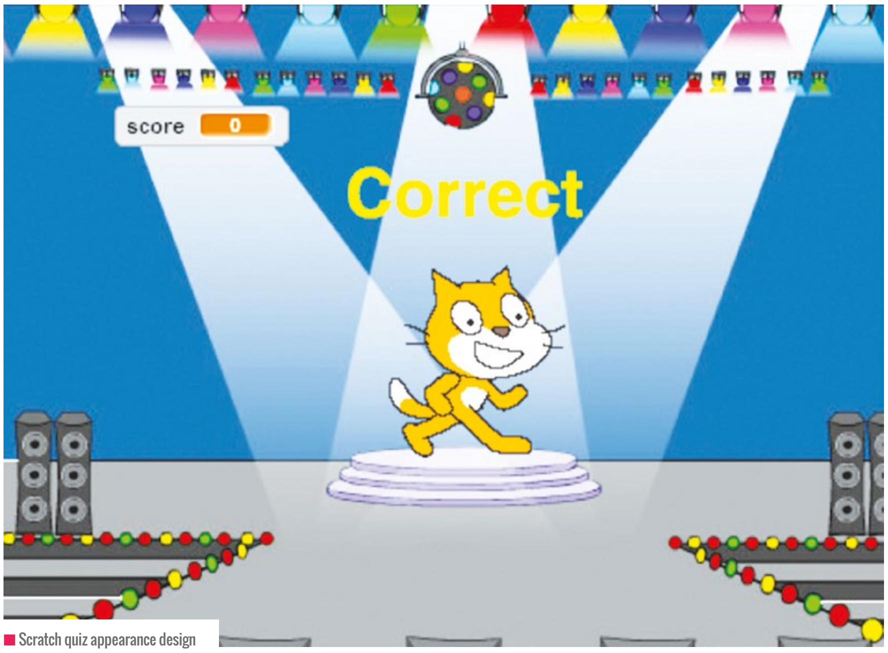
Algorithmic design can be described as thinking through the primary functions of the project – the steps or rules to achieve the primary function that can be turned into code. At primary level, this can include basic data structures, such as variables or simple lists which we might separate as data structure design at a later point.

Appearance design can be thought of as all the aspects of the project that relate to how a project will look and how the user will experience it. User interface design is a big part of this. It might also incorporate screen and physical artwork, and primary teachers may wish to incorporate aspects of the Art curriculum into appearance design.

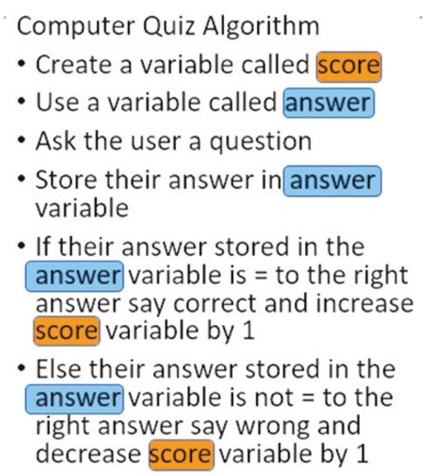
Structural design includes any structural design and technology that we need to include in a project. If we're programming physical objects (like buzzers and motors), we'd include electronic wiring planning as part of structural design.

In a simple quiz, the job of making a quiz question produce the right answer and keep a score belongs to the algorithmic design process. What type of questions the programmer chooses to test, how the quiz is introduced, the user experiences and the chosen effects once the user has got the answer right or wrong belong more generally to appearance design, although they might also necessitate elements of algorithmic design.

When creating an animated animal using the Crumble programming language and servo motors, it's easy to delineate between all three design types. Appearance design involves what choice of animal and what artwork we'll use to make our creations interesting. Algorithm design and subsequent programming involve thinking through what we want the servo motor to do and how we want it to interact with a button or distance sensor. Structural design involves how to wire these in such a way that every component works together – what materials we'll use to mount the servo



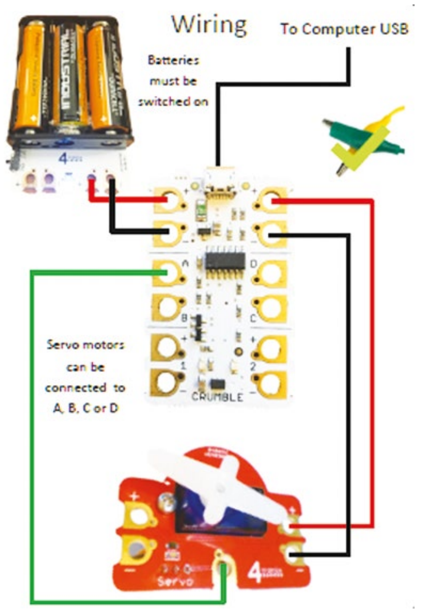
Scratch quiz appearance design



Computer Quiz Algorithm

- Create a variable called score
- Use a variable called answer
- Ask the user a question
- Store their answer in answer variable
- If their answer stored in the answer variable is = to the right answer say correct and increase score variable by 1
- Else their answer stored in the answer variable is not = to the right answer say wrong and decrease score variable by 1

Scratch quiz algorithmic design


Wiring
To Computer USB
Batteries must be switched on
Servo motors can be connected to A, B, C or D
CRUMBLE

Animated animal structural design

and artwork on and how these will be attached.

There's always going to be overlap between all three design areas. Some appearance design choices will necessitate algorithmic design changes. Some structural design choices will limit or extend appearance and algorithmic design, and vice versa. A pupil may be more proficient in one area whilst lacking in skills in one or both other areas. I have a pupil whose algorithmic design is outstanding but who struggles to think about the needs of the people who may use his program. Developing appearance design and the empathy necessary to think about the user are one of his computing targets this year.

Appearance design

As in all learning situations, the teacher may choose to allow greater choice and independence in one aspect and provide greater guidance or support in others. A rounded curriculum would hope to draw on programming projects from a wide variety of genres and give pupils a chance to develop all three aspects of design planning. Currently, due to a focus on algorithmic design, I think it's fair to say that appearance design has taken a back seat in many screen-based programming projects.

In Years 4 or 5, I do a short Crumble project, which involves thinking through the traffic lights sequence in detail before turning this precise algorithm into code on our Crumble micro-controllers and traffic light accessories. It would be possible to mount the traffic light, hide the wiring and concentrate on the appearance design, but to be honest in this project it never feels like it needs this – algorithmic design and the wiring part of structural design are enough for this project. However, the animated animal mentioned in the previous section would be significantly poorer without the appearance design elements.

My Year 6 pupils are currently finishing their independent Scratch platform games. Appearance design and algorithmic design are both important to such a visual programming outcome.

When programming a counting machine with Year 5 this year, the beauty and elegance of it was wrapped up in the elegance of the algorithm, and appearance design is a much smaller part of the project. When we adapt this to create a countdown timer for our teachers then appearance design becomes a lot more important.

I'd summarise by saying that if the project has a significant visual aspect then time spent on appearance design is time well spent. If the beauty of the project is primarily in the algorithmic complexity then appearance design can take a back seat. If a project needs structural design then time spent developing this can help develop pupils' engineering thinking.

Where to start?

Do you have a favourite programming project that you currently do with your pupils? Does it have a significant visual element? If you do, why not give pupils time to think through appearance design. Ask questions such as who's this program for? Encourage them to identify a user group they're writing their programme for. Could they find someone who typifies that group and ask them what they'd want from the type of programme they intend to write?

Primary teachers don't need to separate elements of design and wall them in their own planning space, but it definitely helps to consider algorithmic, appearance and structural design issues before the project. (HW)

DESIGN IS CROSS CURRICULAR

Primary domain competency for algorithmic design generally rests within the Computing Science domain, although in some instances Maths and Science knowledge may be important. Appearance design shares domains with a host of creative disciplines such as Art and Graphic Design, depending on the materials and techniques advocated. The primary domain competency of structural design is Design and Technology. This doesn't mean Computing doesn't have valid competency in these areas, but a good educator might take advice from these areas as well as Computing when developing programming design.

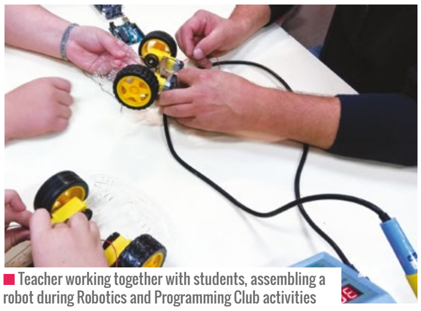
Animated animal appearance design

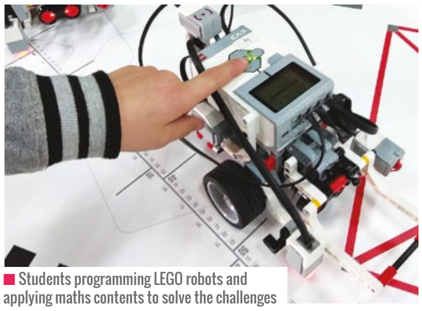
Phil is a Computing Inspector/Advisor working for Hampshire Inspection & Advisory Service and CAS DfE appointed Computing Master Teacher. He was involved at the drafting stage in creating and refining the new Computing Curriculum through the BCS and CAS. He currently teaches Computer Science in two Hampshire schools.
He's a contributing author to Compute-IT KS3 Scheme of work, author of *How to teach primary programming using Scratch* and soon to be author of *Crumble Creations*, helping teachers to develop computing-flavoured STEM projects. His code-it online resources are the sixth most used primary resources in the UK.

NEW TIMES DEMAND NEW EDUCATIONAL APPROACHES

Today, we're living on the cusp of a new industrial revolution, coined the Fourth Industrial Revolution (IR4). This should be the trigger to make schools change and offer new educational contexts to our students, giving them the opportunity to acquire the skills to face the challenges of IR4

STORY BY Marco Neves


Teacher working together with students, assembling a robot during Robotics and Programming Club activities


Students programming LEGO robots and applying maths contents to solve the challenges

Students working collaboratively to solve problems related with a robot that has to go around the Batalha Monastery. An example of cross-curricular content related to history and robotics

Outside schools, new technologies are coming to life everyday. If we pay some attention and look around, we can see: artificial intelligence, 3D printing, robotics, big data, Internet of Things (IoT), virtual reality, genomics, mobile computing, augmented reality, smart cities, and more. And it's quite difficult to foresee, in a short period of time, the new coming ones. In the meantime, what's happening inside schools? Almost the same strategies, the same processes, the same contexts, and the same learning spaces we had in the previous industrial revolutions.

Futurist Gerd Leonhard says that "Humanity will change more in the next 20 years than it has in the last 300 years" and "we may be the last generation that knows what offline means". And do schools still want to be 'offline' from this new reality? A reality where the changes and impacts of this arise at a dizzying speed.

How is it possible to live in the 21st century and still observe situations where instead of trying to perceive what's best to empower students, it's only being considered what's less positive about their performance? We need to develop new learning contexts to change this status quo.

New educational challenges

Regarding some of the skills referred to in literature, we usually find reference to the four Cs (creativity, collaboration, critical thinking, and communication). In my opinion, one C is missing in this context: C stands for curiosity.

But we know that it's not easy to create the right contexts to be able to provide the facilitators that help both teachers and students develop these Cs.

Nowadays, there are different initiatives (both national and European) that give us the technical background support regarding this – lots of them are related to digital technologies such as coding.

But they lack when it comes to collaboration and to communication. They're great in providing the context for developing critical thinking, complex problem solving as well as creativity, but it's difficult to prepare the ground to engage students and teachers in communicating and collaborating.

In my school, we try to provide the necessary contexts to give students the opportunity to develop these skills. We have a Robotics and Programming Club **bit.ly/craeb** (this year it was considered one of the best three in Portugal), and we encourage and help students to take part in initiatives such as 'Code Week' and 'Apps For Good', Astro PI, and CanSat.

But to be able to close the circle around all the Cs, the eTwinning project (**etwinning.net**) is clearly the best pedagogical environment as it embraces all the conditions to enhance all the educational power that allows our students to develop not only 'technical skills' but also the so-called 'soft skills'. These skills are nowadays considered so important to make sure we have flexible and adaptable professionals in a world where we're not sure what the so-called 'future jobs' will be.

eTwinning project - MORE

The main goal of MORE was to enhance knowledge in STEM by building apps, programming robots and Arduinos, and prepare students for the new demands regarding both technical issues and soft skills.

The aim of MORE was to spread knowledge among national networks of teachers who are specialised in specific subjects. The objective was to increase students' participation in learning and to help them be better prepared for their future jobs.

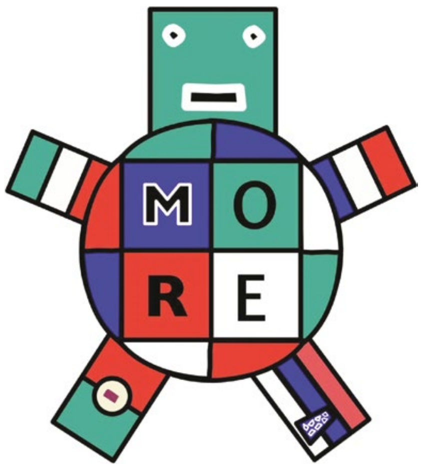
There were four schools involved: Portugal, Italy, France, and Slovenia. Each school was responsible for one area:

- Portugal (Agrupamento de Escolas da Batalha): apps development
- Italy (I.I.S. "N. Pellegrini", Sassari (SS)): robotics building and programming
- France (Lycée Saint Cricq, Pau): developing and programming Arduino projects
- Slovenia (OŠ Preska, Medvode Eslovénia): worked as a beta tester of all the products developed during the project.

All the partners shared their knowledge and expertise in order to:

- Increase the development of devices to record scientific experiments
- Build digital devices to help younger students practise mathematical notions
- Produce Learning Objects (educational resources) related to mathematics and science
- Produce educational games with devices like robots and microcontrollers.

A team within the project was responsible for monitoring the evolution of the project and for evaluating its quality



and impact. They created checklist sheets to register all this information. A survey was applied to evaluate each mobile resource, and interviews with some of the participants were also recorded. Additionally, the main impacts of the project were registered regarding how all the participants improved their competences in different areas. The learning objects produced during the project are available on the project's web page.

Concerning the activities, educational games for mobile devices were created for kindergartens and primary schools, such as an animal quiz, multiplication exercises, temperature, volume, length, area units conversions, and usual geometric figures surface calculations.

Students also created electronic systems controlled by Arduino boards to measure soil humidity and temperature, air temperature for plants, and to measure the sound level in a classroom. Other activities were related to LEGO Mindstorm robots, such as a football field with an automatic scoreboard (controlled by sensors and Arduino boards), where the robots were control by smartphones.

In order to transfer knowledge to the partners of the project about their specific subject, students created tutorials using videos, slide shows, photos and written documents. All these materials were shared and are available for further use on the project's Twinspace page (**helloworld.cc/2JPZAaU).**

As a result of participation on the project, students acquired new skills and competences not only related directly to the technologies but also and mainly to those that are more difficult to address, such as: coordinating with others, communication, creativity, and critical thinking. In this project, we can state that the traditional notion of 'teacher' and 'student' was taken to a higher level because teachers and students have taught other students and teachers, and we've all learned with each other.

Just a conclusion

Within a framework based on the development of 'multi cross-subjects' projects, STEAM can play a very important role by acting as an anchor to the development of projects that can embrace all types of subjects. Using this methodology, schools are creating meaningful contexts for students to be prepared and aware of the 'unknown impacts' of IR4. Amongst all this 'unknown', one thing we know: our students need to be very adaptable, flexible, creative, lifelong learners, and tremendously curious. "Change won't wait for us: business leaders, educators and governments all need to be proactive in up-skilling and retraining people so everyone can benefit from the Fourth Industrial Revolution," says Alex Gray in *The 10 skills you need to thrive in the Fourth Industrial Revolution* (**helloworld.cc/2KymDrV**). So, we have the obligation to create the models and contexts to allow it to happen, otherwise we'll have a generation with a skills shortage for the new demands of the labour market, and that will become a big problem to society. (HW)

Marco is a Computer Science teacher and ICT Coordinator in Batalha, Portugal. An enthusiast of new technologies applied to learning environments like robotics, virtual reality, coding, IoT and 3D printing. Teacher trainer for ICT in education since 2002. Responsible for developing, coordinating and supporting educational partnerships with European countries related to technological innovation projects. Educational technology consultant. Ambassador for the Scientix Project (**www.scientix.eu)**. Microsoft Innovative Educator Experts 2017-2018. Member of Google Earth Outreach Trainer Network.

LORNA ELKES CAS MASTER TEACHER

TO BOLDLY GO INTO THE WORLD OF VR

Although VR is still a costly investment, it offers far more than just gaming-style opportunities for learning

As the technology revolution gathers speed, more and more 'cool' devices and accessories are arriving in schools, including using Virtual Reality (VR) headsets. Following a highly successful visit from one of the touring 'Google Pioneer Expedition' teams, we've invested in our own set of VR devices. From the Pioneer Programme, we'd already seen the potential excitement of pupils and engagement in learning – being able to travel to the past or different planets. The simplicity of use was also key, and staff were up and running with the equipment after a very brief 30-minute training session.

Anyone attending BETT would know that there are several suppliers to choose from, and after some debate we selected the REDBOX VR setup (**redboxvr.co.uk**). They arrived in their own bespoke mobile storage cases, complete with router, and couldn't be easier to set up and use. Although initially it was the classes with more tech-confident staff that used them, we've since trained pupils in Year 5 to support staff with leading VR expeditions in class, and they're now a regular attraction during themed, school exhibitions and open evenings, each time led by students.

Where did we go?

Of course, our early expeditions involved visiting places that wouldn't normally be possible, such as the International Space station, journeying through the layers of an exploding volcano, and visiting the depths of the oceans, all of which provided extraordinary vistas that wowed staff and pupils. The clever teams at Google incorporate interesting snippets for each location, and even a short trip provided a wealth of discussion, engagement, and learning.

Once begun, an expedition can be controlled via the main tablet; arrows appear on explorers' screens to direct them to specific areas within the 360° panoramic views, and an array of corresponding 'smilies' materialise on the guide's tablet indicating where each viewer is looking. Conveniently, the pause option on the main tablet freezes the tour and blanks the VR headset view.

> " Using VR has enabled greater understanding of ancient history. The superb Stonehenge expedition enabled Year 3 pupils to explore the landscape as it would have been many millennia ago

So what are the benefits?

Having explored impossible locations, VR also enhanced real visits and contextualised historical locations.

Like many schools, we're very proactive in encouraging ALL pupils to take part in school visits, yet there are still a few who pupils can miss out. Welcome VR. We now regularly use VR to supplement school trips. This enables all pupils to have a minimum experience of the location; receive a guided tour of the main attractions; and, most importantly, enjoy a shared experience with their peers. Resulting activities are of a higher quality and pupils can

A shared learning day with a visiting school, using the headsets in pairs, still maintains the focus and engagement

Our first experience of VR from the Google Pioneer Expedition Programme

always 'pop back' to check on something they missed when walking around in real life. Similarly, using VR has enabled greater understanding of ancient history. The superb Stonehenge expedition, complete with ghostly images of the past, enabled Year 3 pupils to explore the landscape as it would have been many millennia ago, whilst still appreciating the current layout of the stones.

By using the headsets in pairs, we discovered the on-task discussion increased significantly; paired support was greater, ensuring partners had seen the same; vocabulary used was more specific, particularly directional and descriptive language – having to explain the location of something when someone else is looking; and pupils mastered the art of sharing far quicker than with other activities.

Further still are the hidden benefits of increased responsibility: pupils being trusted with expensive equipment; older pupils rising to leadership roles; and staff recognising that they don't need to always be the experts. (HW)

WHAT TO CONSIDER

- **Use in pairs:** not only does this reduce the number of devices needed for a class, it significantly helps with engagement. Children have to take turns, listen carefully to instructions, and be prepared to discuss and describe the experience.
- **Use a time limit:** use in short burst can be very effective, especially when combined with open questioning and discussion. Current guidance suggests VR is less suitable for children under seven years.
- **Try it out first:** this may seem obvious, but exploring a VR environment on a tablet isn't the same as through a headset. Know the experience you're about to give your pupils. Ensure it's relevant.
- **Consider space limitations:** VR can work just as well if pupils are seated or if they're allowed to wander. Either way, be clear about sudden movements; even when seated, children will turn heads and wave arms.
- **Health considerations:** check pupils' medical details first; children with epilepsy are advised not to use the headset; they can also cause nausea. Using the device outside of the headset can be just as engaging.
- **Share resources:** in the spirit of collaboration, we've arranged visits from other schools, using VR as one of several shared learning experiences.

Lorna Elkes is Deputy Head Teacher at Brookmead School in Buckinghamshire. She has led several subjects including D&T, Maths and Computing.

SUPPORT SEN CHILDREN WITH COMPUTING USING TANGIBLE PROGRAMMING

Tangible programming brings coding to life. You can build, test and debug, all through creating kits that move, shake, wobble and provide immediate feedback

STORY BY Jody Carter @codeyjody

Most tangible programming works by connecting interlocking or magnetic blocks to provide a program that's then run sequentially. Some, such as SAMLabs, are connected via bluetooth and interact with an app to connect blocks together and program, while Cubetto comes with a wooden control panel to set out commands that are then sent to a programmable robot.

As well as teaching children how to code using the tangible programming 'languages', they help develop the skills integral to computational thinking. Wing said that computational thinking involves designing systems and solving problems, and that's exactly what you get with tangible programming.

Can you program the robot to cross the drawbridge and reach the treasure? How many apples can you collect in one move? Is there a way to control my robot with an iPad? Not exactly earth-shattering, but important and interesting ways to teach coding, and computational thinking, to children. Through tangible programming, you can do all of these and much more. By constructing physical real programs, it draws upon prior knowledge of real-world systems. Some are similar to children's games, where blocks have to be passed through windows; others draw upon LEGO and jigsaws, where the individual pieces imply being part of a greater and complete whole.

Programming using tangible programming

Tangible programming allows for some quite advanced programming techniques. Look at Cubetto and the way it introduces functions and subroutines, or Osmo and loops. The more advanced tangible programming languages like SAMLabs provide the functionality to use logic gates and create circuits using motors. Another advantage of tangible programming is that

Use Osmo coding on iPads

Photo: Kaylee Wilsher (www.quantico.marines.mil/Photos/igphoto/2001676807)

Create circuits with Little Bits

Photo: Little Bits

WHAT DOES IT DO?

- Connect interlocking or magnetic blocks to provide a program that's run sequentially
- Bring algorithms to life - create real-life algorithms and 'programs' that do things in the real world
- Touch, feel and see your program moving, blinking or bleeping across the room
- Excellent at teaching the fundamentals of coding and computational thinking

there's no smelly code – having redundant or extraneous code makes your code smelly. With tangible programming, these extra code blocks become obvious and often lengthen the time it takes for your program to complete a task. Why send your Code-a-pillar round the houses (literally) when you can take the shortest and most direct route? With Osmo code, if you take too many steps or go an indirect route, the app tells you and you need to start again.

Kwon, Kim, Shim, and Lee (2012) researched learning gains of first-grade students in a treatment group using tangible coding compared to a control group using only a computer-based coding program (i.e., Scratch). Participants in the treatment group made fewer errors in coding and achieved higher levels of programming tasks.

Tangible programming makes debugging even more fun than when you're trying to find the proverbial needle in a haystack in the 500 lines of Python you've just written. In debugging, you can step through code, replace blocks one at a time or isolate code to run independently. It's also there right in front of you, ready to be undone and rebuilt with your own hands.

Supporting SEN children

According to the CAS #include SEN toolkit, it's important to provide efficient methods of engagement and expression (as well as support). So how does tangible programming provide this? It's physical; you have to interact with it to get it working. It brings the algorithm to life and makes programming concepts less abstract. Many tangible programming kits are colour-coded or use symbols to negate the need to learn complicated syntax. Cubetto are colour-coded and physically represent the process involved. It also facilitates cognitive and fine motor skill learning. Bers found that tangible programming has the following developmental benefits:

- Socio-economic (collaboration, teamwork)
- Emotional development (perseverance, self-esteem, self-efficacy)
- Cognitive development (sequencing, logical and computational thinking)
- Fine motor skills (building, manipulating blocks, using materials)

Cubetto Kit: control a robot to help tell stories

Photo: Primo Toys

SAMLabs STEM kits

Photo: SAMLabs

> TANGIBLE PROGRAMMING GIVES IMMEDIATE FEEDBACK; YOU CAN SEE WHETHER IT'S WORKED OR NOT

The vocabulary of programming is still used: algorithm, program, loop, function are used regularly and children are still able to talk confidently and knowledgeably about their programs. Tangible programming gives immediate feedback; you can see whether it's worked or not. It also avoids those frustrating moments when nothing happens at all, as something generally happens, so it's easier to discern if it's been successful or not.

Scaffolded learning

Tangible programming can act as scaffolding between the virtual and the real world, and helps make the concepts less abstract. It negates the requirement to learn syntax, other than that of placing the blocks in order (which the colour-coding and symbols can help with). Most tangible programming systems operate using flow of control, meaning that the program runs sequentially from one block to the next. Using colour-coded blocks and/or symbols means that children can have success with the blocks without even having to read or write. Programming can also be done in a more natural environment, away from screens and possible distractions; this give more flexibility as to where and when lessons can take place. Tangible programming also lends itself to collaborative work – often the kits are big enough to be easily shared and therefore children can still use a paired programming approach.

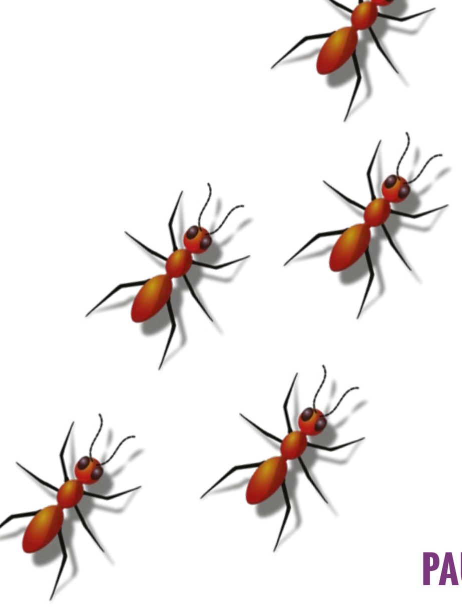
Some considerations

It should be noted here that tangible programming kits can be expensive. They can also be quite fragile, small and easily lost, and in certain cases can only be used for one purpose (a kit that costs £195 is the same price as a Chromebook, for example). One kit can cost upwards of £500 – along with the iPad or PC that's required to run the app, the total cost could reach £1,000, albeit that the device may already be acquired. Others have been designed and produced for home use as toys rather than for educational use in schools, so the need to make them low cost wasn't a requirement.

Another consideration is that children can have difficulty in switching from one interface to another, such as from tangible programming to a virtual one with a screen (Giannakos, 2017). To demonstrate a progression of skills and to enable children to be able to access coding environments such as Scratch or Kodu, it's possible to see a progression of systems and interfaces, beginning with Code-a-pillar, moving to LittleBits and finally SAMLabs, for example. However, this would be extremely costly. (HW)

	CODE-A-PILLAR	CUBETTO	OSMO	LITTLE BITS	SAMLABS
Cost	£49.99	£195+	£49-189	£79-1,800	£129-999
Ease of use	Interlocking USB blocks	Wooden blocks slot into control board and sent to robot	Magnetic blocks snap together; used along with app	Magnetic blocks snap together	Connect blocks via app
Programmability	Sequences	Sequences, loops and functions	Sequences and loops	Sequences and MakeyMakey	Sequences and electrical circuits
Resources	barefootcas.org.uk	www.primotoys.com/education	my.playosmo.com	littlebits.cc/lessons	helloworld.cc/2JLezms
Further	helloworld.cc/2KspK4y0	www.primotoys.com	www.playosmo.com/en/coding	littlebits.com	uk.samlabs.com

What kits are available? A wide range of different tangible programming kits are available that cover a broad range of ages and abilities

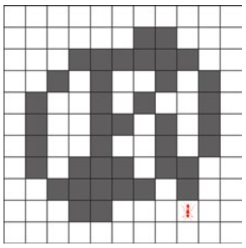


PAUL POWELL CURRICULUM LEADER FOR COMPUTING

LANGTON'S ANT

Implement a simple algorithm with big Computer Science connections

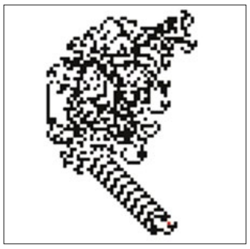
When doing my A Levels, a friend and I became a little obsessed with Langton's Ant. We were playing with this simple program using Pascal, although you can implement it in any language. The idea of the ant is very simple, but it displays some complex emergent behaviour.



Here's the ant after 200 steps

An ant stands on a grid of white squares. It looks at the square it's stood on and acts as follows: if the square is white, the ant paints it black, turns right 90° and moves forward one square; if it's black, the ant paints it white, turns left 90° and moves forward one square. It then repeats the process. If it hits the edge of the grid, it stops.

The ant creates small logical patterns to start with and then a chaotic mess. We called this 'fungus'. Around 11,000 steps in, the conditions are just right, and the ant gets into a repeating pattern that makes it head off steadily in one direction, or build a 'highway' as we called it.



After initial chaos, a highway forms

Coding the ant

Creating Langton's Ant from first principles is quite easy, but it's also an interesting challenge for those newer to code. First, you need to be able to represent the squares. If you're using a graphical language, you'll need some kind of drawing canvas, a function to plot pixels in a given colour and a function to read the colour of a pixel. If you want to do this in text mode, you'll need a 2D array.

We now have a grid that can represent the black and white squares, so let's move onto the ant. The ant has a position (x and y) and a direction (abstraction). It needs to be able to turn left, turn right and move forward. Turning left and right is easy: we just modify the direction. Moving forward is trickier as it depends on the direction (we need some selection to handle that). Once we can solve these smaller tasks (decomposition), we can put it all together to make the ant move a single step. A loop can then be used to repeat the behaviour and complete the code.

ANTS ARE TURING COMPLETE

Langton's Ant is in fact a 2D Turing machine. If you encode data onto the initial grid using black and white for ones and zeroes, you can get the ants to do binary addition, or any other calculation that it's possible for a regular computer to do. It would, of course, take a little longer to design and run your programs.

Further challenges

Once the ant is complete, it's easy to think of extensions. How about adding more colours for more complex rules like turning 45°? What about multiple ants? Would a 3D ant produce the same emergent behaviours? We should, of course, have been doing our A Level coursework. Speaking of which, I must get back to my marking... (HW)

Paul Powell is Curriculum Leader for Computing at George Mitchell School in East London. He also co-leads the Waltham Forest Hub.

MAKING LEARNING HAPPEN

How learning in a hackspace augments and supports learning in a classroom

STORY BY Brian Balmer

Hackspaces, hackerspaces and makerspaces come in many different sizes and guises. They may be just a few tables and chairs at the back of a library, or a few thousand square feet filled with all kinds of equipment. At all of them, though, things are always made and learning always happens. Also, in each space, you'll find a real sense of community, where collaboration occurs naturally and creativity flourishes.

I recently attended a Picademy, became a Certified Educator, and was fortunate to meet the co-inventor of the Raspberry Pi, Pete Lomas. He described how he had always made things from an early age. His creations usually involved cardboard, as that was freely available. Through his inventions, he learned about gravity, basic physics and material technology. This same methodology of learning through making happens at every hackspace. All the learner has to bring to the makerspace is the maker mindset.

The maker mindset in a hackspace is the belief that it's physically possible to transform an idea into a physical object. As Pete Lomas found, this doesn't require a lot of high-tech equipment. Give any group of makers in a hackspace some LEGO, cardboard, arts and craft supplies, and then add a Raspberry Pi or BBC micro:bit to the mix, and fairly quickly ideas will start to flow about what should be made from the kit they've been given,

From a teacher's point of view, I've witnessed an exploration of science, technology, engineering, art and maths. These STEAM subjects are the ones considered the most important for equipping any learner for life in the 21st century.

Perhaps just as important in the whole process is the high degree of computational thinking and planning that often occurs in even the most basic of projects.

In the larger hackspaces, there's also the facility to learn new skills such as woodwork, 3D modelling and printing, electronics, soldering, coding, sewing and photography.

A hackspace is a place that promotes the time to be creative

Start with a project

Many of the people who come to a hackspace have some idea of what they want to get out of it. For example, at our hackspace, we have a growing number of children who originally just came because of the free Code Club. Fairly soon, though, they start to explore the other equipment and resources available. Some have then learned to solder so that they can build circuits. They then go onto learn how to use Arduinos, Raspberry Pis and BBC micro:bits to interact with those circuits. This means that the coding they originally came to learn now has a purpose and their interest in coding increases. The impetus for learning has come from their desire to make and create their project. They're not driven by a curriculum or assessments. Their only form of assessment is whether the thing they've made works or doesn't work. If it does work then their next step is usually how to make it better.

Being surrounded by other makers is a good thing, since one is able to share ideas, skills and knowledge. The downside, though, is that there are many other things to distract your focus and attention. This is when having a time-based project is very useful. The Pioneers programme organised by the Raspberry Foundation is an excellent source for ideas and inspiration. The programme is a digital-making challenge for 11-15 year olds in the UK and Ireland for teams of up to five individuals. Each team has a mentor who'll help them with their chosen project.

At our hackspace, for example, our small team of two intrepid Pioneers are currently been given the challenge of responding to a zombie outbreak. They've to decide what materials they could use and how they're going to use them. They're currently using a Raspberry Pi to control a model brain, which will alert them with sights, sounds and a Twitter message when a zombie appears. To do this, they've had to learn how to use the GPIO pins on the Raspberry Pi, how to code using Scratch and Python, and how electronic circuits work. Taking into consideration that they're only halfway through their project and had previously only a limited experience with physical computing, they're currently on a steep learning curve.

Start with a project

A common criticism levelled at classroom learning is that learners don't have the time in a tightly packed curriculum to explore areas outside of that curriculum.

“ THE MAKER MINDSET IS THE BELIEF THAT IT'S PHYSICALLY POSSIBLE TO TURN AN IDEA INTO A PHYSICAL OBJECT

CODING AT HACKSPACE

The makerspace movement is a global phenomenon. There are currently 2,015 spaces around the world, and that's just the ones that call themselves a hackerspace. You can find these at wiki.hackerspaces.org. Each space, large or small, is a community of makers, learning and sharing skills with each other.

A hackspace, though, is a place that promotes the time to be creative. Part of being creative is also learning by failing. James Robinson, the Picademy Lead Trainer, used the acronym FAIL in his introduction to the Picademy training. “FAIL is simply the First Attempt In Learning,” he said. The users of a hackspace will rarely see their prototype work perfectly first time. Within the maker community that the learners find themselves, though, failing is sometimes a good thing. Through the act of failing, the learner is able to analyse what went wrong and how their next prototype will solve that problem. The mindset that they take back to the classroom is one that includes problem solving and resilience.

How to get involved

Find out where your local hackspace is located and when it's open, then just turn up. There will always be someone willing to show you around the place. Most hackspaces run different events that would welcome volunteers, so you might want to consider this as an option. For example, many run Code Clubs and Raspberry Jams. They're also often looking for volunteers through the STEM Ambassador programme. However you choose to experience your local hackspace, you'll leave with lots of ideas to take back to your classroom. It might even inspire you to set up your own hackspace.

Brian is Hackspace Director at Leigh Hackspace, FE Lecturer, RPi Certified Educator and STEM Ambassador.

A high degree of computational thinking and planning occurs in even the most basic of projects

FLOW CONTROL IN PRIMARY PROGRAMMING

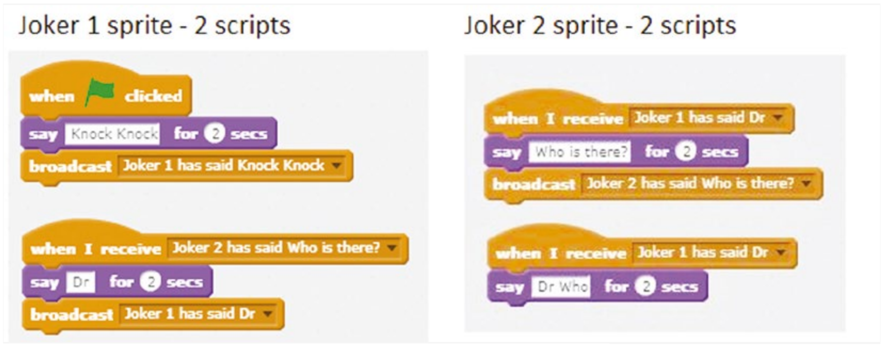
In primary programming, do learners need to understand the flow of control of their program?

STORY BY Jane Waite, William Marsh, William Lau and Jon Chippindall

The English KS1 and KS2 Computing programmes of study don't mention the need to teach pupils the difference between programs where there's a single script that runs one command after another as opposed to programs with multiple scripts all running at the same time. Nor does it mention teaching about coordinating action across such scripts, but tools such as Scratch include commands that encourage learners to implement these features.

If we're teaching programming using simple programmable toys, such as Bee-Bots, there's just one script of code. Nothing can run in parallel on the single device, there's nothing to coordinate, no coordination.

When we look at ScratchJr, concurrency can be seen in several ways. The most obvious is probably when a second character is added, and each character is just 'doing its own thing', each perhaps started by the 'on green flag' trigger command. Time can be used to try and coordinate action, but this is tricky as there's no specific way to control how long a command takes to run. Broadcast using coloured envelopes, the 'send message' command, clearly supports managing coordination across scripts.


```
Joker 1 sprite - 2 scripts

when [flag] clicked
say [Knock Knock] for (2) secs
broadcast [Joker 1 has said Knock Knock]

when I receive [Joker 2 has said Who is there?]
say [Dr] for (2) secs
broadcast [Joker 1 has said Dr]

Joker 2 sprite - 2 scripts

when I receive [Joker 1 has said Dr]
say [Who is there?] for (2) secs
broadcast [Joker 2 has said Who is there?]

when I receive [Joker 1 has said Dr]
say [Dr Who] for (2) secs
```
■ Here two sprites are telling a simple joke. Broadcasts manage the coordination Image: Jane Waite

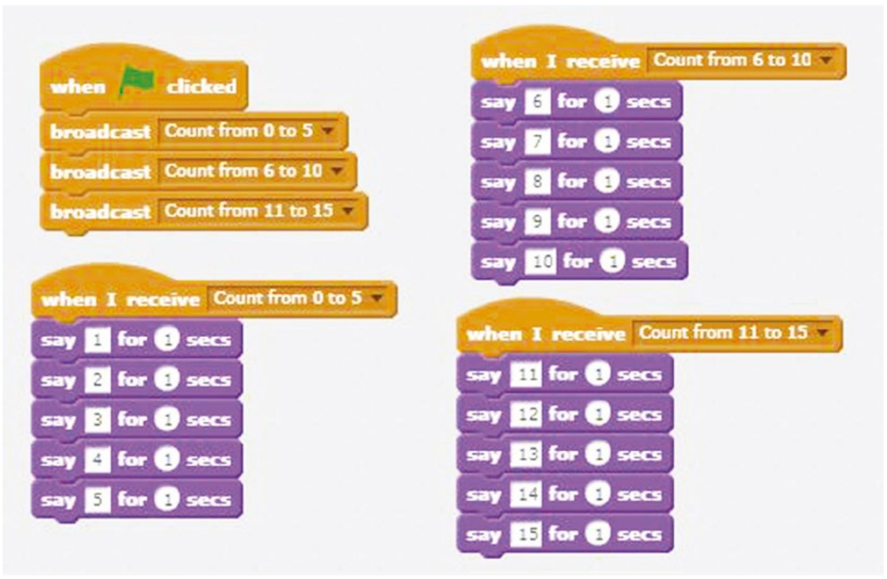
In computer science 'speak', in a block-based language such as Scratch, we might call these running blocks of code scripts; we might call scripts running at the same time as 'concurrent' or 'running in parallel'; we might call coordinating the action across scripts of code as managing 'dependency' or 'synchronising' activity. In this article, we'll use the terms 'concurrency' for scripts running all at once and 'with coordination' for managing dependency.

Further scripts can be created by using the ontouch or bump trigger. In recent research on the use of ScratchJr, using the send and receive message and on bump triggers came in the bottom five of the least popular commands used in over 7.5 million projects, but no report was made on the average number of characters used, nor how often multiple scripts were coded for each character.[1] But we suspect that many young children create programs with more than one character each 'doing its own thing', so they may be writing concurrent programs.

In Scratch, multiple sprites are common. Recent research showed the average number of sprites per project was 5.6, with an average of 17 scripts per project. Broadcast was used in only 30% of projects and 'broadcast with wait' only 4%.[2] How many projects used wait to coordinate across sprites wasn't reported. It seems that primary pupils are writing concurrent code, and many may be expected to create programs that require coordination across sprites.

Scratch was recently reported by The Royal Society[3] as being the most popular primary programming language used in England. Therefore despite the English programme of study not requiring us to teach about concurrency and coordination in primary school, the products we're using to teach programming have such features within them and many learners are using them.

The next question is, should we explicitly guide learners on the use of such features or just let them use them and hope they don't develop misunderstandings or bad habits? Research from a team in Israel found that concurrency and coordination

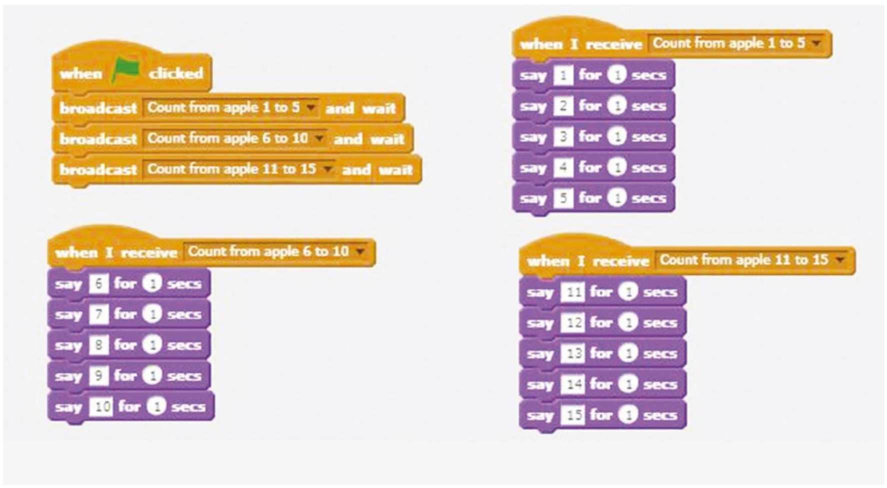


```
when ⚑ clicked
broadcast [Count from 0 to 5 v]
broadcast [Count from 6 to 10 v]
broadcast [Count from 11 to 15 v]

when I receive [Count from 0 to 5 v]
say [1] for (1) secs
say [2] for (1) secs
say [3] for (1) secs
say [4] for (1) secs
say [5] for (1) secs

when I receive [Count from 6 to 10 v]
say [6] for (1) secs
say [7] for (1) secs
say [8] for (1) secs
say [9] for (1) secs
say [10] for (1) secs

when I receive [Count from 11 to 15 v]
say [11] for (1) secs
say [12] for (1) secs
say [13] for (1) secs
say [14] for (1) secs
say [15] for (1) secs
```

What will this set of four scripts do? Predict and then run the code. A copy is at: scratch.mit.edu/projects/198118383 Image: Jane Waite

were two of the hardest concepts for KS3 learners to understand in Scratch and recommended careful teaching to help overcome this.[4] Teachers in English schools, anecdotally, report that learners find these two ideas very hard to understand too. In the ScratchMaths[5] resources, broadcast is taught using an unplugged activity where children call out the lines of 'The Grand Old Duke of York', responding to hearing the previous line in the familiar nursery rhyme.

So how should we teach and guide use of concurrency and coordination? Children understand that they can do more than one thing at once – they can sing at the same time as they jump, at the same time as they wave their arms about. But do we draw their attention to this idea? Do we give 'doing more than thing at once' a name? What vocabulary do we use to talk about coordination? Do we have objectives to meet for these concepts, and how do we assess outcomes?



```
when ⚑ clicked
broadcast [Count from apple 1 to 5 v] and wait
broadcast [Count from apple 6 to 10 v] and wait
broadcast [Count from apple 11 to 15 v] and wait

when I receive [Count from apple 1 to 5 v]
say [1] for (1) secs
say [2] for (1) secs
say [3] for (1) secs
say [4] for (1) secs
say [5] for (1) secs

when I receive [Count from apple 6 to 10 v]
say [6] for (1) secs
say [7] for (1) secs
say [8] for (1) secs
say [9] for (1) secs
say [10] for (1) secs

when I receive [Count from apple 11 to 15 v]
say [11] for (1) secs
say [12] for (1) secs
say [13] for (1) secs
say [14] for (1) secs
say [15] for (1) secs
```

What will this set of four scripts do? Predict and then run the code. A copy is at: scratch.mit.edu/projects/198118383 Image: Jane Waite

SCRATCH AND OTHER EVENT-BASED PROGRAMMING LANGUAGES INVITE THE EARLY USE OF CONCURRENCY

When we play apples and pears with a parachute, children wait until their trigger word is shouted and then run under the parachute to find a new place. In PE, when working in cannon, they wait for the proceeding action before performing their part. In singing, they wait for the preceding line when singing parts. Learners seem to understand the idea of coordination in unplugged cross-curricular activities.

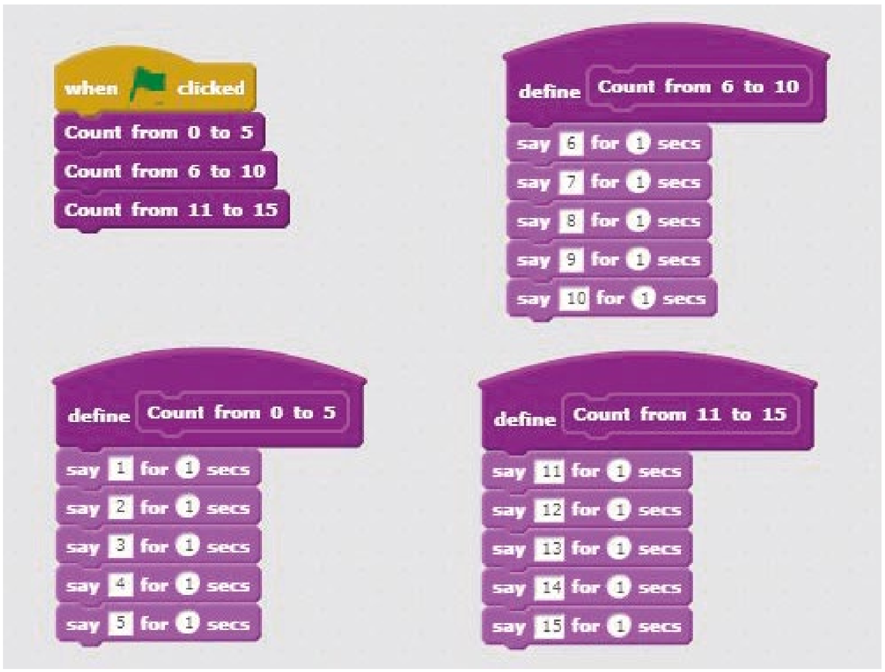
However, do pupils understand that the script running a broadcast command in Scratch does NOT wait for the receiver to finish the action kicked off in the receiving script before it runs the calling script's next command? We suspect not always.

What are the differences between 'broadcast with wait' and 'make your own blocks' (procedures)?

Often Scratch planning introduces broadcast long before make your own block. Make you own block is a way to create a procedure, and procedures act very differently to broadcast (without wait). A script calling a make your own block, a procedure, will wait for that block to finish before moving onto its next step. The calling script stops and waits until the called procedure has finished. Plus, there can be only one make your own block (procedure) with that name. But for broadcasts, there can be dozens of receiving blocks of code with that broadcast name, even when using 'broadcast with wait'.

Usually, concurrency and coordination is an advanced topic, not dealt with until undergraduate studies. For these older learners, procedures and functions are taught early. Should we be doing the same with block-based programming languages?

Scratch and other event-based programming languages invite the early use of concurrency, and following that coordination. Many primary schemes of work exploit this before teaching procedures. It's not clear what this means for learners' progression in understanding of these complex concepts.



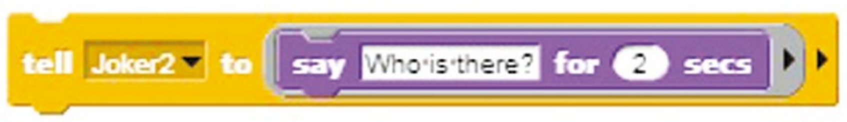
```
when [flag] clicked
Count from 0 to 5
Count from 6 to 10
Count from 11 to 15
```

```
define Count from 6 to 10
say 6 for 1 secs
say 7 for 1 secs
say 8 for 1 secs
say 9 for 1 secs
say 10 for 1 secs
```

```
define Count from 0 to 5
say 1 for 1 secs
say 2 for 1 secs
say 3 for 1 secs
say 4 for 1 secs
say 5 for 1 secs
```

```
define Count from 11 to 15
say 11 for 1 secs
say 12 for 1 secs
say 13 for 1 secs
say 14 for 1 secs
say 15 for 1 secs
```

■ What will this set of four scripts do? Predict and then run the code. A copy is at: scratch.mit.edu/projects/198118383 Image: Jane Waite



```
tell Joker2 to say Who is there? for 2 secs
```

■ Examples of Snap command controlling another sprite Image: Jane Waite

Nor is it clear as to how well primary teachers understand these different concepts, nor how well they understand the misconceptions that may be being taught.

It could be that teaching broadcast before the use of procedures is causing learners confusion and developing misconceptions. It could be that through teaching these concepts carefully in primary we reduce the risk of confusion or minimise the misconceptions secondary teachers then have to navigate.

Just to make things even more complicated, different block-based programming languages implement concurrency, coordination and procedures in different ways. Snap allows any sprite to control any other sprite!

Further work is needed to understand this complex area. We need to create pupil learning trajectories, learning goals, lesson activities and assessment material for teaching and learning of concurrency, coordination and procedures for primary pupils as well as CPD material for teachers to better understand the concepts and possible misconceptions.

But if your learners are using concurrency and coordination, and are having some problems, try the following suggestions, and see how they impact teaching and learning:

“ IT COULD BE THAT TEACHING BROADCAST BEFORE THE USE OF PROCEDURES IS CAUSING LEARNERS CONFUSION

- Teach concurrency using unplugged first, using dance, gymnastics, singing
- Clearly link this unplugged work to creating a program with concurrency
- Teach coordination carefully. Maybe start with wait but move onto broadcast
- Teach broadcast as a means to coordinate action, rather than to decompose a task
- Teach using ‘make your own blocks’ (procedures) to implement decomposition early, such as a simple ‘set up’ script running sprite, or draw a square
- To clearly show that broadcast is about coordinating action, first teach that broadcast should finish a script (with no further commands in the same script) - like the knock knock example. This will help you see a clear flow of control through a program
- Only introduce the ‘broadcast with wait’ command if the learner understands how it works and the difference between it and ‘make your own blocks’ (procedures)
- Look out for creating and receiving the same broadcast within a sprite. Probably ‘make your own blocks’ (procedures) should be used instead
- Avoid broadcasting to multiple sprites (unless the receiving sprites run independently). (HW)

FIND OUT MORE

Why not sign up for one of the CPD sessions with your local CAS Master Teacher? Or ask your local CAS hub for materials from the Diving Deep into Primary Programming Course being used to train Master Teachers in the more tricky bits of teaching programming.

REFERENCES

1 helloworld.cc/2rcjqo0
2 helloworld.cc/2tUFes7
3 helloworld.cc/2jjuWLE
4 helloworld.cc/2Fu7hAW
5 helloworld.cc/2rbD3yc

TEACHERS' PERCEPTIONS OF TECHNOLOGY...

An overview of the different types of teacher we see in schools and how they may perceive technology in their own classroom… which one are you?

WRITTEN BY Sam Palmer

I'm a millennial and, having just come out of the other end of teacher training, I'm in the rare yet fortunate position of having been able to go into many different schools and observe many different styles of teachers in terms of their technology usage. Let's break it down…

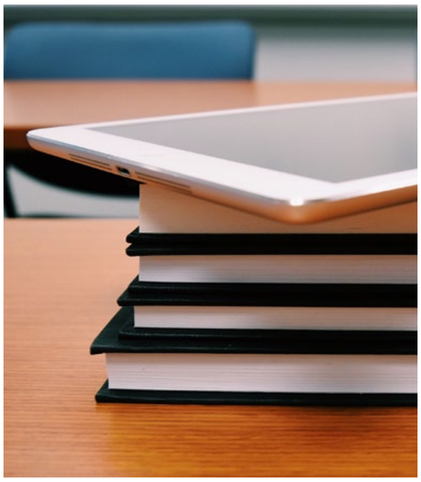
1. The 'tech-savvy teacher'
This teacher is glued to their iPad. They love their interactive whiteboard and go into meltdown when the school's network fails. Technology is integrated into their daily practice and they enjoy showing other teachers new apps, software or websites.

Sadly, this type of teacher is rare. One of the best teachers I've seen using technology was retiring the year after I left, so the stereotype of the young tech-savvy millennial is pointless, simply because it's not always true.

2. 'That's not what it was like in my day'
Every school has at least one. These teachers have been teaching for 35 years, often come out with the phrase 'I was teaching before you were even born' and shy away from those 'thingies' in the corner of their room, or their 'posh white board'.

As a student in their classroom, I once demonstrated how you could use an iPad to record learning in their KS1 classroom and they called me a witch. This is what we're dealing with here. Easy to blow their minds, be careful you don't show them more than one tool at a time.

As they would say… "teaching never used to have all of this technology, so why should I start using it now?" Well…



1. The children
I'd be willing to bet a significant amount of my student debt on the fact that the majority of the children in your classroom have access to technology. In order to be able to effectively reach and teach these children, we need to appeal to what interests them to draw them into their own learning.

2. Awareness
Technology is all around us. I'm sure I don't need to point out the technology around you at this very moment, but it's safe to say technology is taking over the world we live in. It's almost certain that any job a child goes into now will involve them interacting with various technologies. Whether we like it or not, technology is all around us and we, as teachers, need to embrace this.

3. It's fun!
There's nothing better than showing a child something new, seeing their face light up in amazement and excitement.

HAVE A GO… PLEASE!?!

Right. Contrary to popular belief, technology isn't scary. It can't hurt to try something new in your classroom. Why not have a go at using iPads to animate voiceover or spoken language? Or why not use a laptop to create a research presentation? I beg, there are so many benefits to using technology to aid learning and promote a social constructivist classroom environment.

One of my favourite examples of this is my Raspberry Pi Kits. Children are mesmerised by how this tiny piece of plastic, metal and solder is a fully functional computer that can do pretty much anything they can dream of.

Technology is inspiring, and I hope that one day everyone can find a way to successfully integrate it in some way or another to their classroom environment (or at least use our smart boards as collaborative tools and not expensive projector screens… sigh). (HW)

Sam is a final year Primary Education student at Edge Hill with a keen interest in technology in the classroom and the teaching of computing within our current curriculum. Also, he's a Pi Certified Educator (Class of 2016) and from the Isle of Man. <shameless_plug> Want to know more? @SamPalmerEHU </shameless_plug>

MARK THORNBER TEACHER

MATHEMATICAL MUSINGS

Using turtle graphics to investigate repeating decimals

This time, I'd like to return to a topic I looked at in an earlier version of this column[1]: repeating decimals. We turn a fraction into a repeating decimal by dividing the bottom into the top. Here's 4⁄7.

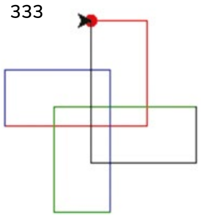
```
        0   .   5   7   1   4   2   8

7   4   .  4 0 5 0 1 0 3 0 2 0 6 0 4 0
```

We stop after six digits because the next remainder of 4 has been seen before so the pattern will repeat. There are actually two sequences here: the sequence of decimal digits: 5,7,1,4,2,8,5,7,1,... and the sequence of remainders: 4,5,1,3,2,6,4,5,1,... In what follows I always use the sequence of remainders. It has the advantage of not containing any zeroes so the pictures are nice.

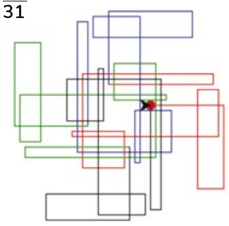
We can turn a sequence into a picture using turtle graphics as follows: take each number in the sequence, move forward by that distance, rotate 90°, then repeat with the next number. To make sure the turtle returns to the start, I use four repeats, in four colours.

Here are three examples and Python program to produce them:

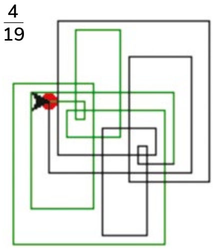
$\frac{41}{333}$



$\frac{14}{31}$



$\frac{4}{19}$



```
mport turtle
num = int(input(“Enter the
numerator\n”))
denom=int(input(“Enterthe
denominator\n”))
remainder = num%denom
remainders = [remainder]
repeating = False
while remainder > 0:
    remainder =
(10*remainder)%denom
    if remainder in remainders:
        repeating = True
        break
    else:
        remainders.append(remainder)
print(remainders)
t = turtle.Turtle()
t.dot(10,”red”)
for colour in
[“red”,”blue”,”green”,”black”]:
    t.pencolor(colour)
    for digit in remainders:
        t.forward(5*digit)
        t.right(90)
input(“Hit return to quit\n”)
```

Some questions for you and your students:

- Why does using four repeats guarantee that the turtle returns to the start?
- How could you modify to return after three repeats or six repeats? (Hint: change the angle.)
- Some fractions return to the start after fewer repeats. 4⁄19 does it in two – that's why we see no red or blue in the picture – they've been overwritten. Can you find a fraction that will return to the start after one repeat? What's going on here?
- What's the purpose of the last line in the program? (HW)

1. *Switched On*, Summer 2016, available at **helloworld.cc/2IeSuiO**

Mark Thornber has been a maths teacher at Durham Johnston for the last 25 years. Mark's first computer was the classic ZX81.

GAINING CONFIDENCE THROUGH HIGH-QUALITY PROFESSIONAL DEVELOPMENT

Robert Lane shares his computer science journey and how first-rate professional development gave him the confidence to persevere

STORY BY Robert Lane

All too often when leading professional development I hear teachers voice their insecurities about technology. Many teachers express doubts about being able to learn what I've learned and then put it to use in a classroom. I assure them that they're not alone in these feelings and I too have had to work through my own insecurities in regards to technology.

A computer science journey

This made me reflect on my computer science journey years ago. I was an above-average user of technology. I led professional development in my district and had spoken at technology conferences, but one thing that I felt was way beyond my reach was computer programming. I'd recently moved into a computer technology position teaching junior high students after teaching elementary students. I knew that with the advancement in technology I needed to provide opportunities beyond a typical computer applications class. I needed to build some computer science into my classes.

Robert leads professional development in Springfield Local Schools. A teacher works to integrate Scratch into his US History courses

I knew computer science was important but I had no programming experience. I was eager to learn how to program and eventually teach it to my students, but thought there just wasn't a way.

Importance of high-quality professional development

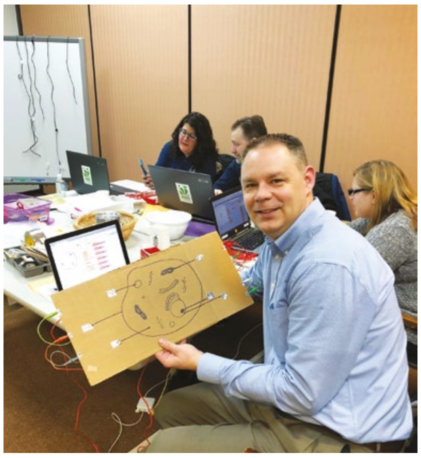
I was then introduced to LEGO Mindstorms and saw the potential it had to offer. Even though I was excited to learn how to program the LEGO robot, I was scared. After being awarded a grant, I was able to purchase 12 sets of LEGO EV3 kits. The school then allowed me to offer a robotics course. I was excited, but I still wasn't a programmer.

By receiving the grant and gaining the approval for a new robotics class, I had no choice but to learn how to program. I signed up for a professional development workshop on programming the LEGO EV3 in ROBOTC through Carnegie Mellon Robotics Academy.

The workshop was excellent. From the start, the instructors gave me confidence in my ability to not only program the robot, but teach my students to program as well. Throughout the 10 hours of training, my confidence grew. In just a few hours, I was able to use what I'd learned to start building my robotics course.

Through this high-quality professional development, I felt a sense of accomplishment. I started to believe I could be a programmer. This grew into a growth mindset that changed the way I approached learning new technologies, such as Scratch and the Raspberry Pi.

When I first started programming the Raspberry Pi, I was again insecure. I didn't have any experience with command line or programming in Python. But thanks to my new growth mindset after learning to program in ROBOTC, I knew that through great professional development I'd be successful at programming the Raspberry Pi too. The high-quality instruction came through an online course and Picademy. Again my confidence grew.


Robert shares his Makey Makey project using Scratch during an Invention Literacy professional development session

First-rate professional development is so important. Teachers are learners too. We need to work through our insecurities and doubts as we build our skill set to better prepare our students for the world. Excellent professional development can do just that. For me, it has taken me from 'I am not a programmer' to 'I AM a programmer!'. (HW)

Robert teaches computer technology and robotics at Springfield High School and Junior High in Ohio. He is a Raspberry Pi Certified Educator and a ROBOTC for LEGO Programming Certified Instructor.

STUDENT-CENTRED LEARNING AN EDUCATIONAL SURPRISE

In the Netherlands, students work independently on final school projects. This manner of working requires an innovative approach: a student-centred learning environment

STORY BY Aad van der Drift & Ramon Moorlag. Students Jelle van Bost, Yochem van Rosmalen and Nathan Guirado

In the Netherlands, students have to carry out a final research project in the last two years of their secondary school education. It's called a 'subject cluster project' because students can choose from a number of subject combinations. They only have to create one subject cluster project, whether it's in collaboration with others or not.

Each student selects a subject cluster in the course of their secondary school education. With this serving as a basis, they pick a topic that's related to a major cluster subject. The grade they're awarded is part of their final examination result.

The student carries out a small research project on their own or in a small group. This serves two purposes: on the one hand the student can gain more theoretical knowledge, and on the other hand the skills of the student are tested, such as the setting up of a research project, experimentation, analysis, description and presentation. Many Dutch schools feature a presentation night for the subject cluster projects.

Jelle, Yochem and Nathan started their research project a year ago – a combination of computer science and biology. They asked if it were possible to develop a hand prosthesis using a few resources and without prior knowledge of the subject matter. Subject cluster projects challenge students to gain more in-depth knowledge of a subject/field of study, so having sufficient prior knowledge of a research project isn't an immediate requirement. In most cases, the role of the teacher is limited to providing advice and support.

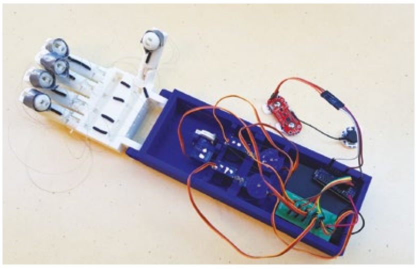
The following detailed answer to the research question was written by the students themselves.

Detailed outline of the project

We started this project with limited knowledge of Arduino software, electronics and 3D design. We also didn't have sufficient knowledge of the motor skills of the hand. The challenge that we faced was to first master the basic knowledge of the above-mentioned components. This turned out to be a long path full of changes, blockades and difficult research work. An example of such a blockade was the fact that we made a wrong decision with respect to the use of an electric motor. Whilst we had initially opted for a stepper motor, a servo motor eventually turned out to better meet our product requirements. Because of a difference in control between a stepper motor and a servo motor, the latter was a better option due to the spinning precision, as a result of which the fingers ended up in exactly the correct position.

At the beginning of the process, we considered which requirements a prosthesis would have to satisfy. Next, we shifted our focus from the functioning to the required components such as sensors, motors, 3D print and wiring in order to keep everything manageable and wearable. In our opinion, the unit had to become a prototype that's suitable for further development. After studying the motor skills, we selected the required components and matching software. We made our choices entirely independently. We ran into many problems

Students built a 3D printed prosthesis, with a lot of control over process and execution


Functional prototype of a 3D-printed prosthesis. Every part was designed by the students. This prototype also supports muscle control

Students Yochem van Rosmalen, Jelle van Bost and Nathan Guirado (from left to right) talking about their functional prototype

Ramon Moorlag, computer science teacher in the Netherlands

INGREDIENTS

The students used the following computer science ingredients independently. Manuals and more information can be found at **github.com/3PMAP**

- TinkerCAD.com
- Arduino Nano
- Fritzing (PCB)
- GitHub.com (code sharing)
- MakerBot (3D printer)
- Myoware Muscle sensor

LESSONS LEARNED

Student-centred learning has taught us to:

- Leave the learning up to the students
- Ensure commitment among students
- Have students formulate their own goals
- Have students establish their own standards of quality
- Allow for mistakes to be made
- As a teacher, learn from the students every day

SUBJECT CLUSTER PROJECTS CHALLENGE STUDENTS TO GAIN MORE IN-DEPTH KNOWLEDGE OF A SUBJECT

with this design, the majority of which we resolved independently. We contacted our supervisor Ramon Moorlag with respect to two problems. The first prototype was jammed since the fingers didn't close and open correctly. In order to improve the mechanical aspects, we designed a simple printed circuit board using Fritzing (freeware software). We gradually also discovered how we could measure muscular activities using the Myoware Muscle sensor. The hand prosthesis would be controllable from the biceps. The five servomotors would subsequently be controlled by means of Arduino Nano and own written code.

Final assessment

Ramon Moorlag, computer science teacher and supervisor for this subject cluster project: "The result created by these students is admirable. I saw them struggle with this project for a whole year. Throughout the process, I saw a development that went from highly simple to more and more complex, with the students constantly demanding more and more of themselves. The project was ultimately awarded a 9.8 (in the Netherlands, we award grades with 10 being the maximum), which means that the project was nearly perfect. This high grade is the result of, among other things, the intensive use and understanding of complex tools. As regards this hand, the students developed something from nothing. They designed all of the phalanges, palm of the hand and the casing of the mechanics using the TinkerCAD application (freeware). That was mostly a process of trial and error. The basis for the manner of working was provided by the computer science subject. Students are taught to work with robotics databases, web technology and algorithms, and there's a lot of room for personal interests. Coding is also part of the curriculum.

Entrepreneurship is one of the components we want our students to focus on. Because of this, these students were independently able to approach experts inside and outside the school in a timely manner whenever their project came to a halt. Particularly noticeable is the fact that they thought up their own process and execution. For example, they examined the functioning of muscles and how to measure this by means of a sensor in a hospital that employs one of their parents.

The assessment of the subject cluster project paid particular attention to the manner of working and research. After all, it's possible for students to be unable to overcome blockades whilst still having carried out a good research project." (HW)

Aad and Ramon are committee members of the i&i, Computer Science Teacher Association and advocates of innovation in computer science education in the Netherlands. Ramon teaches computer science at the ds. Pierson College in 's-Hertogenbosch, the Netherlands. Jelle van Bost, Yochem van Rosmalen and Nathan Guirado are students at the ds. Pierson College. The students are enrolled in the pre-university education programme.

TANGIBLE PROGRAMMING IN EYFS & KS1

Ever wonder how best to teach programming to Reception and Key Stage 1 children in a tangible manner - read on to find out more!

STORY BY Helena Cheung and Eleni Vasileiadou

"Teachers are currently presented with a plethora of educational technology resources that lack pedagogical instruction…there's an urgent need for practical guidance for teachers on what pedagogy should be employed to maximise investment and minimise risk." **(source: helloworld.cc/2r8alcf)**

The quotation from the *Royal Society of Literature Review* on pedagogy in teaching has highlighted a much-debated question of which pedagogy will bring better results and maximise children's progress. We investigated the impact of different approaches (pedagogies) and progression in Reception and Key Stage 1 classrooms for one specific area.

Cubetto was chosen as a context for delivering the pedagogies. For over three months, we trialled the Cubetto in two schools across three year groups to find out how best to teach tangible programming to Reception and Key Stage 1 children.

The main aim of the trial was to find out how to make coding tangible, where tangible means clear enough or definite enough to be easily seen, felt or noticed and to be clearly grasped by the mind (*Collins Online Dictionary*).

Children worked in mixed ability pairs. We tried six different pedagogical approaches. Bee-Bot activities were run alongside the Cubetto activities. We compared our results and examined which approach helped the children move on with their learning. The pedagogies we used were suggested by Jane Waite's scaffolding continuum to 'guide tasks and activities'. The findings of the present trial are summarised in Tables 1, 2, 3 and 4.

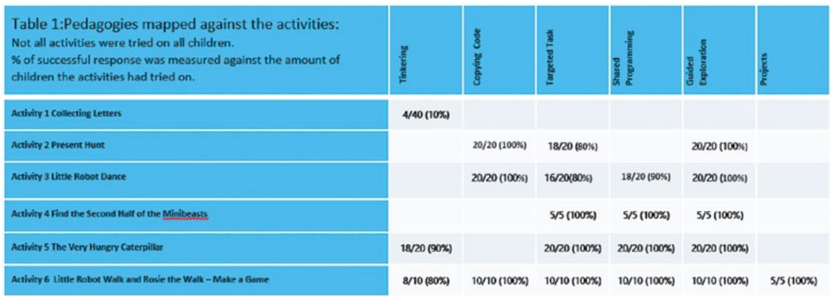


Table 1:Pedagogies mapped against the activities: Not all activities were tried on all children. % of successful response was measured against the amount of children the activities had tried on.	Tinkering	Copying Code	Targeted Task	Shared Programming	Guided Exploration	Projects
Activity 1 Collecting Letters	4/40 (10%)					
Activity 2 Present Hunt		20/20 (100%)	18/20 (80%)		20/20 (100%)	
Activity 3 Little Robot Dance		20/20 (100%)	16/20(80%)	18/20 (90%)	20/20 (100%)	
Activity 4 Find the Second Half of the Minibeasts			5/5 (100%)	5/5 (100%)	5/5 (100%)	
Activity 5 The Very Hungry Caterpillar	18/20 (90%)		20/20 (100%)	20/20 (100%)	20/20 (100%)	
Activity 6 Little Robot Walk and Rosie the Walk – Make a Game	8/10 (80%)	10/10 (100%)	10/10 (100%)	10/10 (100%)	10/10 (100%)	5/5 (100%)

Table 1: Pedagogies mapped against the activities

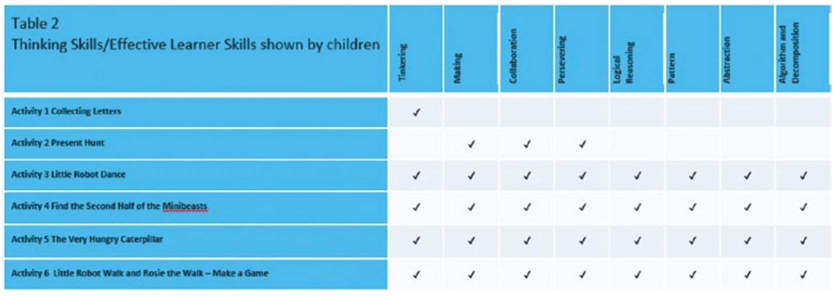


Table 2 Thinking Skills/Effective Learner Skills shown by children	Tinkering	Making	Collaboration	Persevering	Logical Reasoning	Pattern	Abstraction	Algorithm and Decomposition
Activity 1 Collecting Letters	✓							
Activity 2 Present Hunt		✓	✓	✓				
Activity 3 Little Robot Dance	✓	✓	✓	✓	✓	✓	✓	✓
Activity 4 Find the Second Half of the Minibeasts	✓	✓	✓	✓	✓	✓	✓	✓
Activity 5 The Very Hungry Caterpillar	✓	✓	✓	✓	✓	✓	✓	✓
Activity 6 Little Robot Walk and Rosie the Walk – Make a Game	✓	✓	✓	✓	✓	✓	✓	✓

Table 2: Thinking skills/effective learn skills shown by children

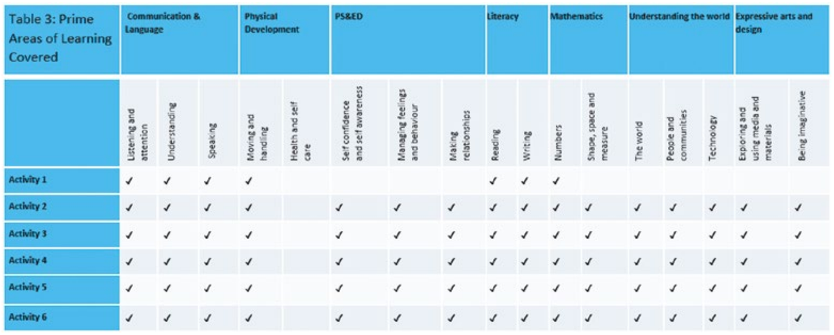


Table 3: Prime Areas of Learning Covered	Communication & Language			Physical Development		PS&ED			Literacy		Mathematics		Understanding the world			Expressive arts and design	
	Listening and attention	Understanding	Speaking	Moving and handling	Health and self care	Self confidence and self awareness	Managing feelings and behaviour	Making relationships	Reading	Writing	Numbers	Shape, space and measure	The world	People and communities	Technology	Exploring and using media and materials	Being imaginative
Activity 1	✓	✓	✓	✓					✓	✓	✓						
Activity 2	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Activity 3	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Activity 4	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Activity 5	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Activity 6	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 3: Prime areas of learning covered

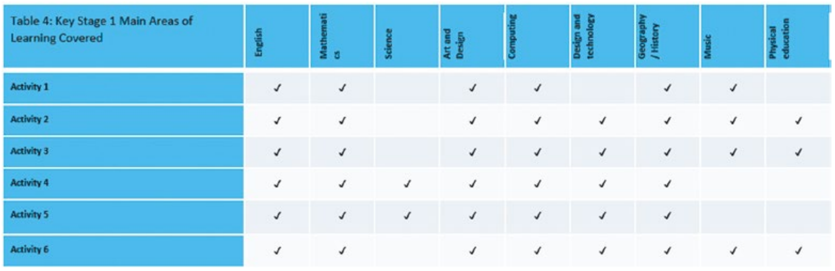


Table 4: Key Stage 1 Main Areas of Learning Covered	English	Mathematics	Science	Art and Design	Computing	Design and technology	Geography / History	Music	Physical education
Activity 1	✓	✓		✓	✓		✓	✓	
Activity 2	✓	✓		✓	✓	✓	✓	✓	✓
Activity 3	✓	✓		✓	✓	✓	✓	✓	✓
Activity 4	✓	✓	✓	✓	✓	✓	✓		
Activity 5	✓	✓	✓	✓	✓	✓	✓		
Activity 6	✓	✓		✓	✓	✓	✓	✓	✓

Table 4: Key Stage 1 main areas of learning covered

What we found out

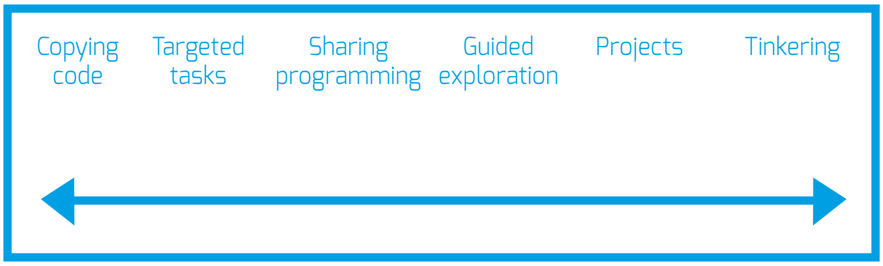
From our trial, we felt Cubetto had made programming tangible by offering a tactile learning opportunity. We found that children learned by participating mentally and physically as they manipulated coding blocks to interact with the Little Robot. They had immediate feedback from the Little Robot when the codes were executed, and there was active thinking in this concrete process. The colour-coded logic blocks drew on children's previous knowledge of colour and colour matching. This minimised barriers posed by children's weak directional concepts, low language ability or the EAL children's lack of language.

The colour-coded logic blocks are symbolic representations of basic Logo primitives. Only forward, left, right and go are used. Having a simple, restricted set of commands helps children to think in a logical manner, developing sequencing and computational thinking in a tactile manner. There are only 16 basic blocks: 14 for basic programming and two light-blue blocks for doing repeat procedures. It looks terribly insufficient at first glance, but after trialling it, the restricted amount of blocks is actually an advantage for learning. When we used Cubetto with Years 1 and 2 learners, we found that we could develop the children's advanced coding skills such as sequence and repeat procedure using the blue blocks naturally. They could see the needs and quickly grasped the skills.

The coding box was a focal point of discussion. Children planned, discussed and negotiated what blocks to put in. This encouraged collaborative work and promoted turn taking. They also learned how to celebrate failure as a chance for more learning. We found children developed effective learners' skills naturally as long as planned scaffolding instructions were provided.

Coding directly on the coding box was different from other common early years/KS1 programming devices in the classroom. Cubetto executed the codes without any intermediate steps for transferring the program. For other early year/KS1 programming devices, children have to plan with a piece of paper and transfer the code to the device, and can't then see the codes after they've typed them in. Mistakes seem to always happen in this transfer process and, worse of all, they don't know what's put in. This makes debugging hard, but Cubetto in trial showed that it could help cut out all the intermediate steps. Both teachers and children knew what was planned and what instructions the

> WE FELT CUBETTO HAD MADE PROGRAMMING TANGIBLE BY OFFERING A TACTILE LEARNING OPPORTUNITY



Continuum of approaches for teaching programming

A continuum of approaches for teaching programming

Kingsgate Primary school children working with a Cubetto. Source: **helloworld.cc/2I4TEgl**

WHAT IS CUBETTO?

The Cubetto Playset is a toy that allows children to code by programming a wooden robot. It consists of a coding board and coding blocks. Children place the blocks on the board to create a program and make the robot move.

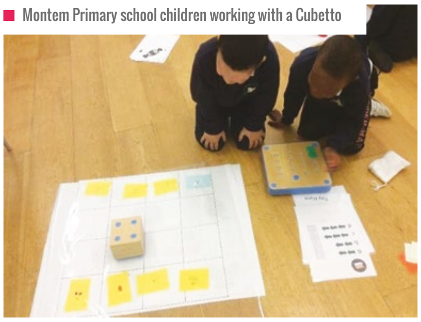
THE SIX DIFFERENT PEDAGOGIES WE TRIALLED WERE

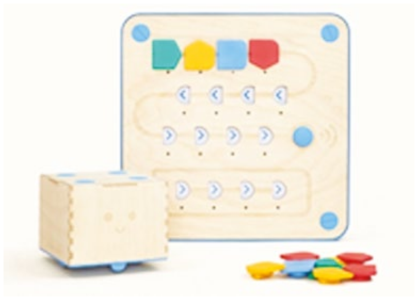
Copy code, Targeted tasks, Shared programming, Guided exploration, Projects, and Tinkering.

MAIN FEATURES OF CUBETTO

- It works with a wireless coding box with a procedure loop
- Planned codes are executed directly from the coding box to the Little Robot
- There are colour-coded logic blocks, which are a representation of Logo primitives
- There are a limited number of blocks

Montem Primary school children working with a Cubetto





A Cubetto Playset

Little Robot was carrying out. The whole process is very concrete to children.

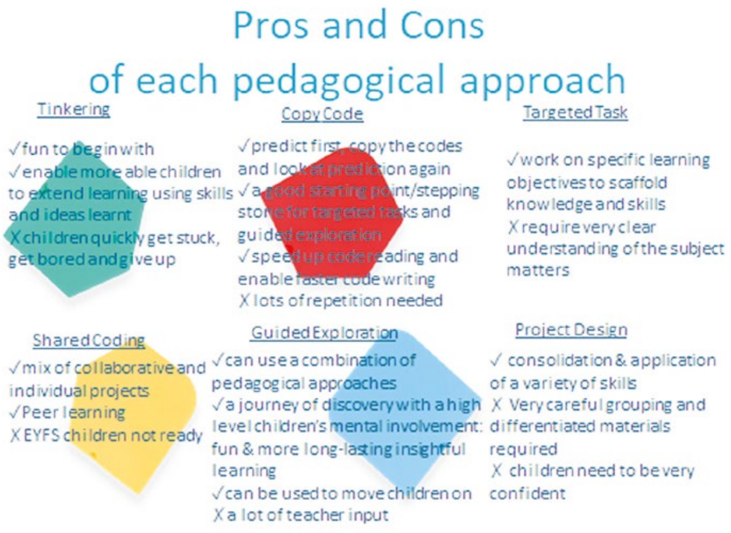
However, Cubetto is just a tool. It's a Little Robot doing what we ask it to do. It can only come alive with teachers' planned scaffolding approaches.

In our trial, children struggled when they were left to tinker without any guided instructions – they got bored and quickly gave up. But when we carefully guided them through, they were more engaged and made greater progress.

However, we found no one pedagogical approach was perfect on its own. We had most success when we carefully planned a blended approach. Children found it hard to tinker because they didn't have any prior knowledge or experience of coding. Copy code also wasn't easy for young children to follow because they had to match the plan representations ↑↑ to the coding blocks, and most found that confusing.

We noticed that a meaningful cross-curricular context was important for any strategy to work. This could motivate children and get them engaged more willingly. In the beginning of the trial, children showed no interest in working at all when they were just asked to move the Little Robot over the mat. But as soon as the cross-curricular context was introduced, they started getting the purpose of learning, and showed great interest and resilience in working through problems.

Our short trial with Cubetto has evidenced that the best pedagogical approach to teaching reception children tangible programming was a blended approach, with guided exploration at the core, because we could support children and provide scaffolding to enhance learning.



Pros and Cons

of each pedagogical approach

Tinkering

√fun to begin with
√enable more able children to extend learning using skills and ideas learnt
X children quickly get stuck, get bored and give up

Copy Code

√predict first, copy the codes and look at prediction again
√a good starting point/stepping stone for targeted tasks and guided exploration
√speed up code reading and enable faster code writing
X lots of repetition needed

Targeted Task

√work on specific learning objectives to scaffold knowledge and skills
X require very clear understanding of the subject matters

Shared Coding

√mix of collaborative and individual projects
√Peer learning
X EYFS children not ready

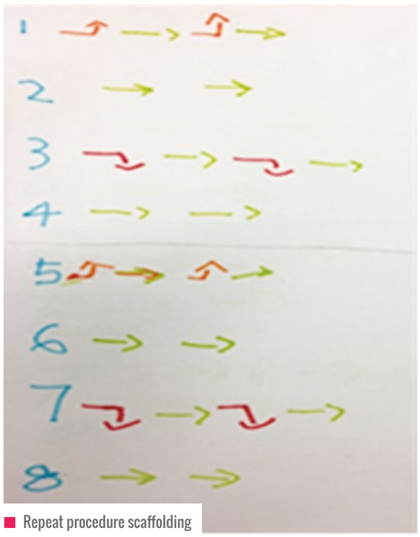
Guided Exploration

√can use a combination of pedagogical approaches
√a journey of discovery with a high level children's mental involvement: fun & more long-lasting insightful learning
√can be used to move children on
X a lot of teacher input

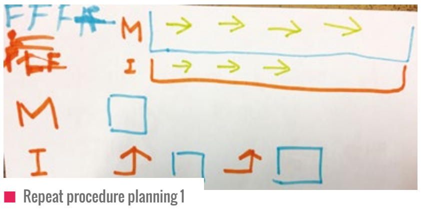
Project Design

√ consolidation & application of a variety of skills
X Very careful grouping and differentiated materials required
X children need to be very confident

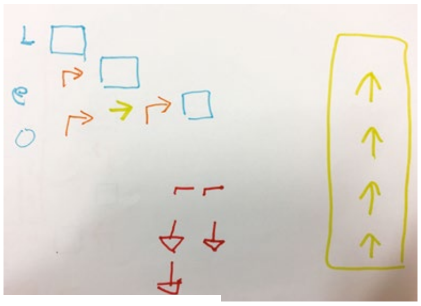
Pros and cons of each pedagogical approach



Repeat procedure scaffolding



Repeat procedure planning 1



Repeat procedure planning 2

After comparing with Bee-Bots, we found that Bee-Bots were an easier start for teaching coding to our Reception children, as they didn't have a coding board and blocks, and everything happened on the actual Bee-Bot. The only difference was that children needed to remember to clear the code before proceeding to a new one. After building the basic skills with Bee-Bots, we then introduced pupils to sequencing instructions. Cubetto was good for providing progression, introducing more repetition, procedures, and applying their skills to make games. The trial has shown that Cubetto could give enough challenge for older children in Key Stage 2. (HW)

ADDING BLOCKS TO SNAP! FOR AI PROGRAMMING

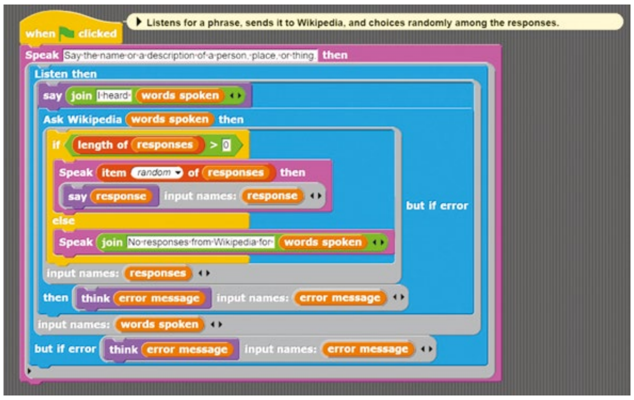
Enabling children to build AI programs

GUIDE BY Ken Kahn

The year 2017 saw a sudden emergence of interest and support for AI programming by children. Stephen Wolfram wrote a chapter in his *An Elementary Introduction to the Wolfram Language* textbook about machine learning. He wrote a blog post "Machine Learning for Middle Schoolers", where he describes the rationale behind the different examples and exercises (**helloworld.cc/2KqeEgB**). Dale Lane launched machinelearningforkids.co.uk, which he recently described in his excellent article in *Hello World 4*. Google launched the first of two AIY projects for voice and vision kits for Raspberry Pi projects (**helloworld.cc/2JJAFFP**). Google also launched the Teachable Machine (**helloworld.cc/2jlxoS0**) and a collection of browser-based machine learning experiments (**deeplearnjs.org**).

The idea that children should be making AI programs goes back at least to the 1970s. I wrote a paper on this in 1977 called "Three Interactions between AI and Education" published in *Machine Intelligence 8* (**helloworld.cc/2KrXQpp**). Seymour Papert, Cynthia Solomon, and others at the MIT Logo Group saw many potential benefits from students doing AI projects. One can learn about perception, reasoning, language, psychology, and animal behaviour in the process of building perceptive robots and apps. It provides fertile ground for self-reflection – students may begin to think harder about how they see and hear, for example. Students may be better motivated because their programs are capable of impressive behaviour. Today, we can add the motivation that students will learn about a technology that's of increasing importance in science, medicine, transport, finance, and much more.

Forty years later, I returned to my interest in supporting AI programming by children. The idea initially was to add blocks to Snap! that used AI cloud services to provide speech input and output and image recognition. Speech synthesis and recognition is now provided free by the Chrome browser. Blocks to speak or listen were created that interfaced to these APIs.



```
when [flag] clicked                ▸ Listens for a phrase, sends it to Wikipedia, and choices randomly among the responses.
Speak (Say the name or a description of a person, place, or thing) then
  Listen then
    say (join (I heard ) (words spoken) ◂▸)
    Ask Wikipedia (words spoken) then
      if <(length of (responses)) > (0)>
        Speak (item (random ▾) of (responses)) then
          say (response)   input names: (response) ◂▸
      else
        Speak (join (No responses from Wikipedia for ) (words spoken) ◂▸)
      input names: (responses) ◂▸
    then (think (error message))   input names: (error message) ◂▸      but if error
  input names: (words spoken) ◂▸
but if error (think (error message))   input names: (error message) ◂▸
```

Entire program for talking to Wikipedia using AI Snap! blocks

Image recognition services are provided by Google, Microsoft, IBM, and others, but only to those who supply an 'API key' in queries. Fortunately, these can be easily obtained and allow for at least 100 free queries per day.

Inspired by Dale Lane's addition of machine learning to Scratch and Google's Teachable Machine, I then added machine learning blocks to Snap!. Currently, only camera input is supported. The addition of audio, images, text, and data is planned. Thanks to Google's efforts on deeplearn.js, this runs fast within the browser by using its GPU. No servers needed.

Sample Snap! AI programs include:

- Speaking with random pitch, rate, voice, and language
- Moving a sprite with spoken commands such as 'go forward' and 'turn right'
- Generating funny sentences in a Mad Libs style using speech synthesis and recognition
- Generating stories using speech synthesis and recognition
- Talk to Wikipedia
- An app that uses speech input and output and image recognition to describe what's in front of the camera
- Training a sprite to move or turn, depending on which way your finger is pointed or you're leaning.

Links to the new Snap! commands, sample projects, and a teacher guide to AI programming in Snap! are all freely available at **helloworld.cc/2jlybSY**. No installation required. We're eager to hear the experiences of teachers who try out any of this, and to answer any questions you have. (HW)

Ken is a senior researcher at the University of Oxford. The work described here was supported by the eCraft2Learn project (**project.ecraft2learn.eu**) funded by the European Union's Horizon 2020 Coordination & Research and Innovation Action under Grant Agreement No 731345.

BLUFFER'S GUIDE TO PLANNING A COMPUTING SCHOOL TRIP

A successful Computing trip can go a long way to engage and inspire your students as well as influence their future study and career choices. In this practical guide, **Alan O'Donohoe** (Computing At School Master Teacher and leader of exa.foundation) suggests suitable trips, activities and venues, and considers how to get the best value from your venture

What's your reason for a trip?

It's fundamental that you establish your prime motivation for planning any school trip long before you start to explore potential dates, venues and other details. With all of the necessary logistics planning, home-school communications, adherence to school procedures and permissions, safeguarding procedures and risk assessments required, even a short trip out of school can quickly escalate into a major undertaking that will make you question why you ever embarked upon the idea in the first place. If you find the whole process daunting or off-putting, it might be better to plan a small trip in the first instance to build up your confidence.

Since there's such a bewildering abundance of trips, activities and venues available, by being clear and absolutely focused right from the start about your intended outcomes as well as your target group, you'll find it easier to make the right choices during the decision-making process.

Suggested motives for a Computing trip

It's possible that you have more than one motive for planning your trip. Sometimes, the motive can be external or forced, such as when you receive an instruction from a line manager to organise a school trip. Here are some typical reasons:

- To widen the appeal of Computing to a particular group of pupils who typically don't select GCSE Computer Science or A Level Computing
- To increase accessibility for all to Computing, boosting opportunities available to pupil-premium groups
- To reward a group of Key Stage 2 digital leaders for their efforts
- To add meaning and context to a particular programme of study, such as cryptography.

Planning for success

Anyone who has ever planned a family holiday will know that with any trip there's a delicate balance to strike between the quality of the experience and the quantity of participants. Planning a successful Computing trip for 14 pupils is a much easier feat to pull off than planning a trip for 180 pupils. If those 14 pupils have previously expressed enthusiasm and excitement in Computing, it's likely to be a more enjoyable experience for all concerned. The problem you run into with a larger group is the increasing likelihood that the presence of unwilling participants will have a detrimental effect on the enjoyment of others. Inevitably, you'll

Vintage adding machine

The National Videogame Arcade, Nottingham

end up planning a trip that's much broader in appeal but doesn't specifically address the needs and interests of everyone attending.

When I was asked by the Assistant Head at my school to plan a Computing trip for 180 Year 8 pupils, I spent a long time exploring the various alternatives. In the end, we planned two different experiences and asked the pupils to select the one that appealed most to them. We offered either a trip out of school (in school uniform) to take part in Computer Science-related workshops limited to the first 80 pupils, or a more general creative, ICT-related, school-based activity day in non-uniform for the remaining 100 pupils. Allowing pupils a degree of choice guaranteed higher levels of success and enjoyment.

Must it take place on a school day?

If you're planning your trip to take place on a regular school day, it might prove to be very popular, purely because you're offering pupils a legitimate opportunity to avoid a lesson they dread. At the same time, you might struggle to attract the pupils you want because they don't want to miss a particular subject they love. Organising a trip during a school break or on a weekend ensures you'll recruit participants with a genuine interest, and the planning can be less stressful since it has relatively little impact on other school activities and doesn't require staff cover.

I've found that the trips I've led outside school times are much more enjoyable and far less stressful. These can also serve as very informative reconnaissance missions for future trips, especially if you're not feeling particularly confident about organising trips for larger groups.

Tunny Gallery, The National Museum of Computing

Choosing the best location to visit

This requires lots of careful thought and consideration, guided by your original motivation for organising the trip. I'd suggest that there's no such thing as a 'wrong location' for a Computing trip, but some venues are inevitably going to suit your needs better than others. It seems incredibly risky to plan a school trip to a venue that you've never visited yourself. So, if possible, I'd

Colossus Gallery, The National Museum of Computing

recommend you organise an initial visit by yourself, possibly accompanied by friends/colleagues or with some younger family members to judge their thoughts and reactions.

Many of the most popular locations, such as the Science Museum (London), Bletchley (Milton Keynes), The Centre for Computing History (Cambridge), The National Videogame Arcade (Nottingham) have dedicated education teams willing to provide advice and guidance. If you give them plenty of advance notice, it's possible they may be able to provide some additional activities or arrange for some guides or volunteers, especially if you're willing to be flexible about your dates.

Planning your journey

When I've spoken to groups of pupils after trips I've organised, they frequently cite the journey there and back as being one of the highlights of the trip. They always seem to appreciate the opportunity to spend some time with friends new and old, sharing fresh experiences together. Where possible, I've tended to plan trips around public transport for long journeys and a minibus from a local taxi firm for shorter journeys. Rail journeys become more affordable with Family & Friends Railcards, and some rail operators offer group travel options; you also have the added option of meeting at a railway station if this is more convenient.

CASE STUDY

In 2013, to help increase the proportion of females choosing to study GCSE Computer Science at our school, I invited a handful of our Key Stage 3 girls to a Stemettes event taking place on a Saturday afternoon in central London. I identified a small group of girls who I thought would benefit from the trip and asked them if they'd like to invite a friend along for company.

After our early morning train arrived in London from Preston, we took a pleasant morning stroll through Hyde Park on our way to the Science Museum to explore exhibits related to Computing. A couple of hours later, following a brief stop-off at Harrods for souvenirs, we headed to the Stemettes event, where our girls had the opportunity to meet a panel of women working in STEM-related careers. We managed to pack quite a lot into one day.

Years later, looking at photos of my exam groups, I realised that almost all the girls who attended our trip chose to study GCSE Computer Science. Some girls forged new friendships on the trip that transgressed age and year group, and later many volunteered as ambassadors for Computing in our school.

FURTHER READING

- I wrote a blog post a few years ago with links to further resources and suggestions: **helloworld.cc/2JHgjgg**
- In the Computing At School community forums, there are regular discussions about the best locations for trips. If you'd like to suggest recommendations, why not contribute to one of these?
- **helloworld.cc/2HHkaO2**
- **helloworld.cc/2KvNvJd**
- **helloworld.cc/2rgqsd6**
- **helloworld.cc/2rdJWyW**
- Miles Berry has created a list of recommendations: **helloworld.cc/2HKT3NZ**

EDSAC Replica Project, The National Museum of Computing

Enjoying the exhibits at Bletchley

RECOMMENDED VENUES

- **Science Museum, London**
- **The National Museum of Computing, Bletchley**
- **The Centre for Computing History, Cambridge**
- **Bletchley Park, Bletchley**
- **The National Videogame Arcade, Nottingham**
- **The National Science & Media Museum, Bradford**
- **The Museum of Science & Industry, Manchester**

Activities on location

Many of the locations teachers recommend are essentially museums with permanent exhibits and collections. While these may be suitable for more mature pupils, the best strategy is to plan an itinerary of activities for your group. Treasure hunts and quiz activities can be planned that require pupils to visit key artefacts and exhibits to extract particular clues or pieces of information, in effect steering them to visit areas of highest relevance and importance. The most successful groups can be rewarded with treats or prizes on the trip home.

By planning for a fairly broad variety of additional experiences during a visit – such as some time for shopping, watching a film related to the topic, or visiting a popular food outlet – you'll be ensuring higher levels of satisfaction and enjoyment.

Collaborate with colleagues

If you can persuade colleagues from another curriculum area in your school to plan a combined trip, this may help reduce some of the legislative and logistic burden for you. I've known schools organise joint Maths and Computing trips to the Science Museum, London and joint Computing and History trips to Bletchley.

Can you help?

If you've led a successful Computing trip, why not add your suggestion to one of the discussions listed above or write an article for a future issue of *Hello World*?

#CASWalkabout

On the last Wednesday of every month, excluding December, a group of Computing teachers meet up at the Science Museum, London for an evening of STEM and Computing-related activity. If you'd like to know more or attend a future event, search for the next #CASWalkabout event promoted online. (HW)

YOUR LETTERS

A world full of possibilities!

Dear Hello World,

In 2017, I attended Picademy in London. It reignited my interest in the Pi after having one on the shelves at home, displayed like a decoration and getting dustier.

The training gave me the confidence to explore what can be done with a Raspberry Pi. The community I belong to now, 'Raspberry Pi Certified Educators', keeps the flame going by sharing projects and offering additional training.

Slowly but surely, I started working on the projects offered with the Cam Jam Edukit 1 and developed by Cambridge Raspberry Jam. This is a starter electronics kit to bring code 'alive' by using LEDs, a button and a buzzer. Discovering the GPIOs (General Purpose Input/ Output pins) opened a world of opportunities; who knew that I'd be planning how to make my home a Raspberry Pi-controlled zone – Internet of Things (IoT), here I go! I even got a 'Voice Kit' for the Raspberry Pi.

Since January, weve been running a Python and Electronics club at Redbridge Central Library. The club, supported by volunteers, introduces children to the Raspberry Pi and basic electronics using the Edukit 1. The participants will progress to use the other kits in the series so they become familiar with sensors and finally get an insight into robotics.

With the aim to inspire others, I led a digital workshop for librarians in January as part of the Society of Chief Librarians roadshows, where I demonstrated how to make a robot antenna.

Picademy have started programming live webinars, and I recently attended one to build a robot with a Raspberry Pi, cardboard and an LED light connected to the GPIOs and coded using Scratch. A simple project to show the capabilities of the Pi. And now I continue my learning with the support of an amazing online communitym and I hope that your readers feel like giving the Raspberry Pi a try.

Maria Reguera
Digital Services and Innovation, Vision Redbridge Libraries

Many thanks for getting in touch, Maria. If anyone else is interested in the work of Picademy, you can read about what's coming up, and tap into resources, at **www.raspberrypi.org/training/picademy.**

Our letters page is a place for you to join our conversation. If you've got a comment, a question or an announcement to share, contact us on Twitter via @helloWorld_Edu or using the #HelloWorld hashtag. Alternatively, email us with 'Teacher Letter' in the subject line (contact@helloworld.cc).

Stretching The Budget

Dear Hello World,

I've really enjoyed your magazine, after a colleague alerted me to its existence. The frustration I have, teaching in an inner-city comprehensive school is that, whilst I love a lot of the ideas that are being discussed – both in your magazine and amongst colleagues – we keep coming up against resource problems. Our school is undergoing another round of budget cuts at the moment, and it seems that it's the interesting ideas around the edges of what we do are first to go.

Raspberry Pi has been something of a godsend, but even so our budget only goes so far. I do wonder if other educators out there could share their tips of not only their ingenious ways to get ideas across to their students, but also innovative ideas that don't involve me having to go cap in hand to my department head to try and get them approved. The kind of conversations where, all too often, both of us know the answer before the chat has even begun!

Keep up the good work!

Name supplied

Snap!, Smalltalk and paradigms

Dear Hello World,

Issue 4 has a thoughtful discussion of programming paradigms. I have, however, a quibbles regarding the claim that object-oriented languages are also structured languages. This isn't true of the first object-oriented programming language - Smalltalk.

Smalltalk has no conditional expressions, instead messages are sent to a "true" or "false" object and the appropriate code fragment in the message is run. Furthermore, there are many more programming paradigms not discussed - such as concurrent, logic, and constraint programming; and message passing. The article correctly states that Scratch is not an object-oriented language and barely a procedural one.

I was glad to see that Hello World recommends Snap!. I strongly disagree with the implication that Snap! isn't a "full-on programming language". Snap! is really Scheme, a Lisp-dialect, with a blocks syntax.

Ken Kahn

Many thanks for your feedback, Ken, and points taken. All feedback, as, always, is very welcome. As an aside, too, you can read Ken's piece on AI Blocks in Snap! on p77.

Join our free online training programme

If you're a teacher, community educator, or volunteer, our **free online courses** will help you to learn more about Computer Science and digital making.

Start a course today: rpf.io/trainonline

Raspberry Pi The Raspberry Pi Foundation. UK registered charity 1129409 #RPiLearn

Q I'M LOOKING TO GET GREATER USE OF MY EV3s INTO MY CODING CURRICULUM - ANY HELP OR IDEAS?

A Depending on the number of robot kits you have, there's nothing like a little competition to get children inspired by a topic. You could have them work in groups and then compete against each other on various challenges, such as tackling mazes, navigating an assault course or sorting blocks, and award prizes for the fastest, most ingenious or even most hacky scripts and bots they can make.

Q A STUDENT ASKED ME WHO 'OWNS' THE PROGRAM THEY'VE PUT TOGETHER IN CLASS. WHAT SHOULD I TELL THEM?

A In short, they do. Unless, of course, your school has a contract with your students that software created by them is your property – this is how employment contracts work.

If you release software, it's important that you provide a licence (even if it's open source), which states how it can or can't be used. There's a summary of open source licences at **helloworld.cc/2Kx30QS**.

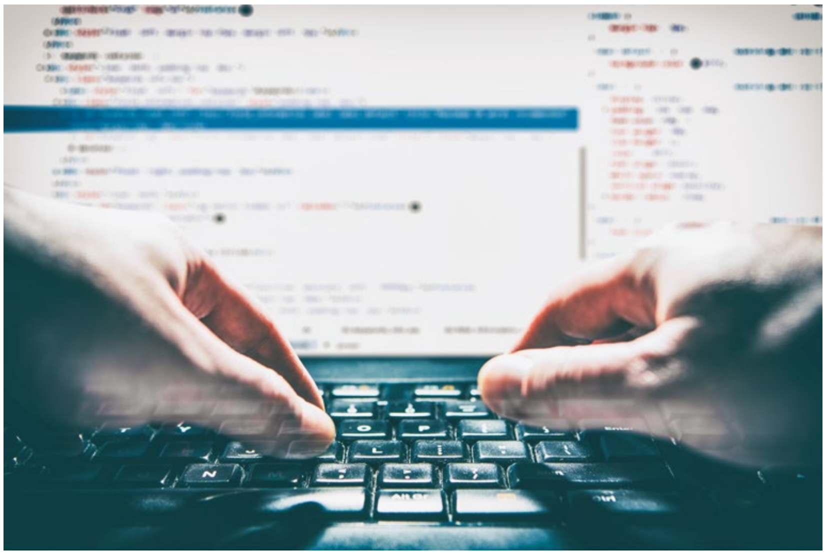
Q DO YOU HAVE ANY TRICKS FOR GETTING THE BALANCE BETWEEN PRACTICAL AND TRADITIONAL PROGRAMMING TEACHING?

A It depends on what you mean by 'practical' and 'traditional' programming, as most programmers would probably consider the two things the same.

However, it's often the case that the awarding bodies for a particular course have a defined set of skills that they wish the students to learn, as well as a fair amount of theory. Delivering this within the time constraints of a school year, while also having students building practical applications, can be tricky.

Where possible, you can try and combine the two. If you need to teach binary to denary conversions, why not have them write little sub-routines that perform the task without resorting to using libraries that do the job for them? Or if teaching about Truth tables, give them an algorithm and have them alter it so that variable values are printed out and the Truth table is generated automatically.

There can be times when it's tricky to find ways to combine the demands of the awarding bodies and keeping up your students' interests, though. You could always encourage them to start developing side projects that they can develop in their own time with a little support from you. Then, if they finish their work early or you've no particular demands for a homework that week, you can get them working on their project.



Q ARE THERE ANY GRANTS OR SCHEMES THAT WE CAN ACCESS IN THE UK TO HELP US BOLSTER HOW WE TEACH PROGRAMMING?

A Last year, the government announced they would be spending £84 million to train up another 8,000 GCSE teachers for Computer Science. How this money will be distributed and in what form the training will come hasn't yet been decided, but it should be announced some time this year.

There's another scheme you might like to look at in the meantime. Rocket Fund (**rocket.fund**) can help schools access the latest technology by connecting them with businesses and their community.

Q IS THERE A LOW- COST EQUIVALENT OF THE NINTENDO LABO THAT I CAN USE IN MY CLASSROOM?

A If it's low-cost physical computing that you're trying to achieve, then micro:bits, Arduinos or the Raspberry Pi are probably the way you want to go. Start simply with a few blinking LEDs or spinning motors, and work your way upward from there.

The Raspberry Pi Foundation has an online course for teachers to help introduce physical computing at **helloworld.cc/2HQem0D**

While something as advanced as the projects for the Nintendo Labo, which all interface with the Switch, might be tricky, you can build a robot buggy with all kinds of sensors for under £20. An example project with a kit list can be found at **helloworld.cc/2w6NZC4.**

Q AT WHAT AGE CAN I REALISTICALLY START TEACHING CODING TO MY PRIMARY SCHOOL STUDENTS?

A The short answer is any age. But it will very much depend on the children in your classroom, their prior experience and the resources you have available to you.

You can start them off with some unplugged activities, to get them used to some key concepts and some of the perplexing terminology that computer scientists and programmers use. Even very young children can understand simple conditionals such as:

If it's warm outside, just wear a T-shirt Else were a jumper as well

Or looping logic such as when eating a pot of yogurt:

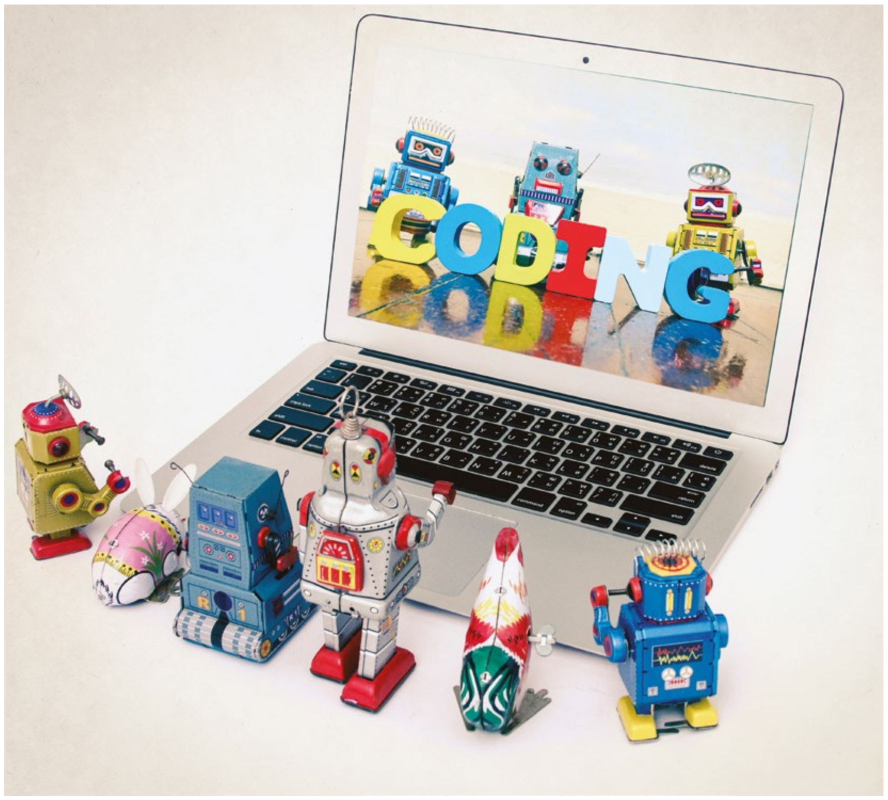
Repeat until the pot is empty Scoop yogurt onto spoon Put spoon in mouth

You can find some amazing unplugged activities online, such as the ones available at **code.org/curriculum/unplugged** aimed at ages four and upwards.

Then, of course, if you have the computers, you can move you students on to block-based languages such as Scratch. Again, this is appropriate for almost any age, and so long as your students are comfortable using a mouse, they can start making small animations, stories and games.

The Raspberry Pi Foundation has a wide range of resources for teaching Scratch at **helloworld.cc/2Ia7bDY**, but there are plenty of other resources out there for many different block-based languages.

Finally, if and when you feel they're ready, you could look at introducing some basic HTML skills to get them used to writing code in text, and then move them on to a text-based language such as Python or JavaScript. This last step is up to you and the progress your students have made. Many children don't move on to text-based languages until they arrive at a Secondary school, while others are already pretty confident coders.



Q DO YOU HAVE ANY IDEAS FOR HOW I CAN CROSS OVER MY STUDENTS' INTEREST IN TWITCH AND GAME STREAMING WITH LESSONS I CAN TEACH IN MY CLASSROOM?

A Twitch certainly is popular at the moment, but obviously you can't have students watching it in your classroom or streaming from their school computers.

You could try a slightly retro approach to this, though. Why not use some simple screen recording software to have students record their programming and IT tasks, talking through what they're doing as they work? If they're making Scratch games, for instance, they could even then play the game while recording a typical Twitch like commentary.

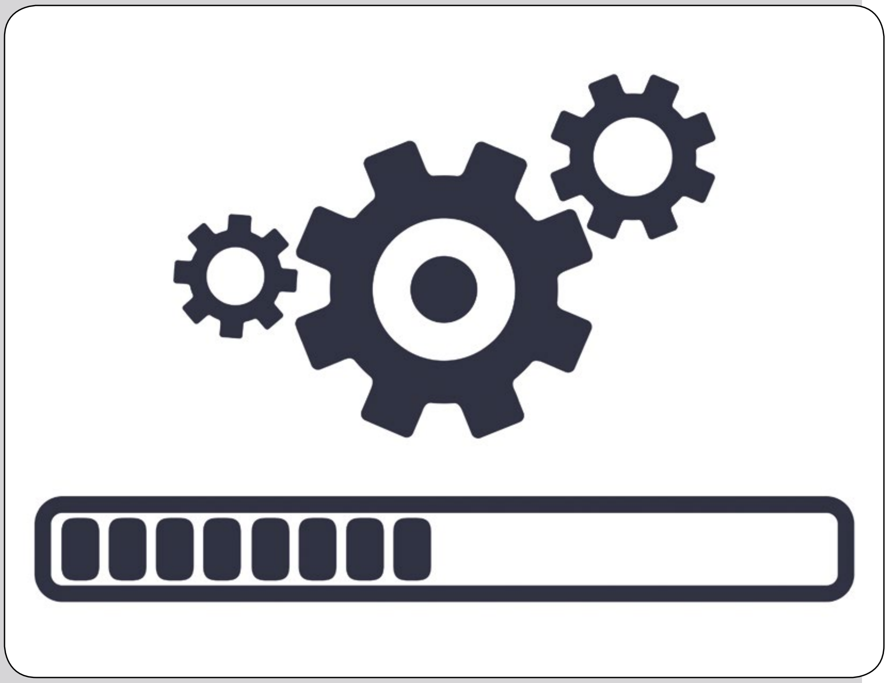
You could then host these videos on your internal network, and encourage them to watch each others' work, and add comments to a text file somewhere.

Q CAN YOU RECOMMEND SOME SAFE RESOURCES FOR PYTHON MODULES?

A Python modules can be categorised as core and 'everything else'; core modules are those released with Python and installed by default, 'everything else' cover those modules which need to be installed and are generally created by community developers.

The Python Package Index (**pypi.org**) is the main repository for Python modules, and the vast majority of modules are available here, but with any software you should review what you're installing before doing so.

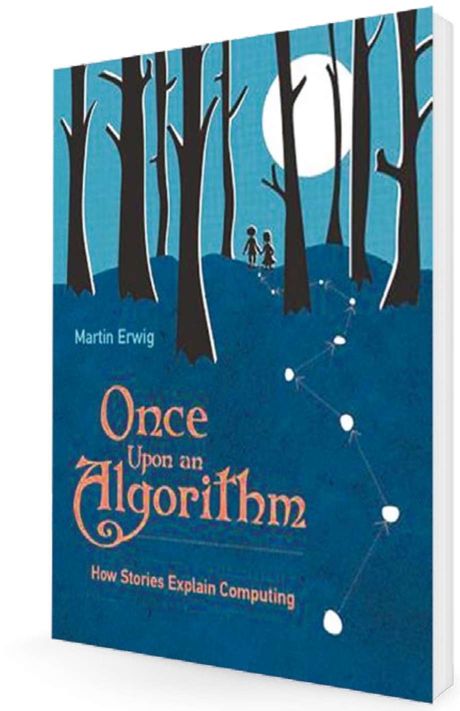
The Python wiki lists some useful modules (**helloworld.cc/2KypbGp**), and the Python Module of the Week (**pymotw.com/3**) blog gives really good reviews and tutorials on widely used modules.



ONCE UPON AN ALGORITHM HOW STORIES EXPLAIN COMPUTING

Exploring the links between Computer Science and storytelling in fairy tales, books, and films

INFO **BY** Martin Erwig | **PUBLISHER** The MIT Press | **PRICE** £16.49 | **ISBN** 9780262036634 | **URL** **mitpress.mit.edu/books/once-upon-algorithm**



When running an algorithm, things change, over time, according to a plan, just as in a story, and so this book uses stories and films (Hansel and Gretel, Sherlock Holmes, Indiana Jones) to explain algorithms and computation.

Hansel and Gretel execute an algorithm to get home from the forest in (literally) a number of steps using a representation (data structure) of a path home (pebbles), but the book shows a number of ways an algorithm could have gone wrong for them, discusses time complexity (it introduces quadratic run times by imagining Hansel going home for a new pebble at each step) and space (resource) constraints (suppose Hansel runs out of pebbles because his pockets weren't deep enough?).

The next section discusses representation and data structures in more detail, using examples from Sherlock Holmes (though Hansel and Gretel still appear when they use breadcrumbs as path markers but the birds use them as food!). Lists, arrays, and trees make an appearance here, as do stacks (Hansel and Gretel's pebbles) and queues (I've never thought of inheritance rules, as in *The Hound of the Baskervilles*, as an example of a priority queue).

More modern examples appear with Indiana Jones, concentrating on searching with the introduction of binary trees and tries, and then moving on to sorting.

Groundhog Day: repetition, repetition...

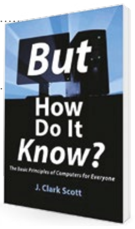
The second section of the book covers how algorithms can be described in a form suitable for a computer (i.e., in a programming language), starting with an exploration of how the song 'Over the Rainbow' can be represented: as a score or using a grammar.

Groundhog Day is used to introduce the idea of repetition and whether we can automatically detect programs that don't halt (the Halting Problem), then *Back To The Future* introduces recursion. Recursion can be one of the harder topics in Computer Science, and the approach taken here, although wordy, gives a number of different examples and strategies for dealing with it, which could be useful (I've never come across insertion sort expressed recursively before).

The final section covers the concepts of types and abstraction using the Harry Potter books as a framework.

At times hard work, this book is full of useful insights and examples to use in teaching a range of topics from Computer Science. (HW)

BUT HOW DO IT KNOW? THE BASIC PRINCIPLES OF COMPUTERS FOR EVERYONE



INFO **BY** J. Clark Scott | **PUBLISHER** John C. Scott, Oldsmar, FL 34677
PRICE £12.95 | **ISBN** 9780615303765 | **URL** **buthowdoitknow.com**

The author starts with a NAND gate, then builds NOT and AND gates with 1 and 2 NANDs, then with 4 NANDs to build a single bit memory, and from these simple components builds 8-bit registers, then connects these registers using wires (buses), decoders to select particular rows and columns in a matrix of registers (memory), and by p66 has shown you a computer's storage!

He then builds a CPU with NANDs, first building OR and XOR gates with 3 and 4 NANDs, then building shifters, adders, and comparators for an Arithmetic Logic Unit, then a Control Unit using a clock and stepper to Fetch and Execute.

Final sections cover iteration and selection (branching instructions), input/output and software (including OSes), finishing with a philosophical discussion deciding the answer to 'But How Do It Know?' is 'It doesn't know anything!'

A software emulation including a simple compiler is available, and to see a computer implemented with mainly NAND gates go to blog.kevtris.org/?p=62. (HW)

COMPUTER SCIENCE WITH - SNAP! BY EXAMPLES

INFO **BY** Eckart Modrow | **PUBLISHER** emodrow@informatik.uni-goettingen.de | **CE** Free (but donations welcome via PayPal) | **URL** **emu-online.de/ComputerScienceWithSnap.pdf**

This book is a fantastic resource for anyone interested in Computer Science: I'd imagine it's aimed at A Level and beyond, but some of the projects would interest students of any age.

Snap! is used to implement solutions to an incredible range of Computer Science projects, introducing concepts such as object orientation by prototyping in an influenza simulation example.

After a short pedagogical section entitled 'CS and Media Studies', Snap! itself is described and the influenza example (for experienced users) given. The projects that follow start simply (a swimming game, solar system simulation, and a Caesar cipher implementation), but go on to cover topics like SQL, pattern recognition (faces, number plates, barcodes), physical simulations, image and sound processing, computer algebra and automata!

Each project has a list of the concepts it introduces, such as duplicated objects, communication via messages, local and global variables for the swimming game, and there are additional exercises at the end of each section. (HW)

ESSENTIAL READING

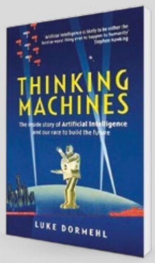
Three recent books about artificial intelligence and machine learning

THE MASTER ALGORITHM

BY Pedro Domingos
PUBLISHER Penguin
PRICE £6.99
ISBN 0141979240

An engaging, non-technical introduction to different approaches to machine learning, covering rule-based systems, neural nets, Bayesian methods, nearest neighbours, and deep learning. The book concludes with a discussion of how these approaches might be combined to create the eponymous 'master algorithm'.

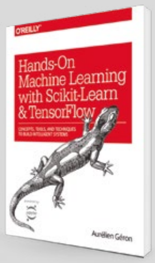
THINKING MACHINES



BY Luke Dormehl
PUBLISHER WH Allen
PRICE £8.16
ISBN 075355674X

An historical survey of the development of AI from the 1950s onwards, tracing (and explaining) the changes in approach that researchers adopted, and including some great character sketches of those involved. Dormehl also shares something of his expectations for where AI will head next.

HANDS-ON MACHINE LEARNING WITH SCIKIT-LEARN AND TENSORFLOW



BY Aurelien Geron
PUBLISHER O'Reilly
PRICE £23.77
ISBN 1491962291

A more technical, and more practical, introduction to key machine learning tools for Python programming. The approach is through narrated worked examples, building a good repertoire of skills for machine learning applications, and developing a working grasp of the underpinning technologies.

WHY CODING IS LIKE COOKING

Dan Fisher discusses how his gastronomical interests led him to a deeper understanding of what coding is

STORY BY Dan Fisher

Introduce young learners to Computer Science through something they already enjoy. Like eating cake. Image courtesy of Alex Bate (Raspberry Pi Foundation)

I've always loved food: making it, discussing it and, of course, eating it. But it's only recently that I've given some thought to the similarities between preparing food and writing a piece of software. In this article, I'll be touching on my early experiences of cooking and how the lessons I learned can be applied to programming. The reason I decided to write this piece is that I think it's important to find different ways of engaging learners who might not get excited about Computer Science through the usual channels. Drawing comparisons between the two disciplines will hopefully give you some food for thought (sorry, couldn't resist!) to engage the hitherto-unreached demographics in your own place of learning.

Scoping the project

Cooking was a source of mystery and awe to me as a child. The kitchen would be full of loud noises and wonderful smells, and my mother would have a tattered, flour-covered recipe book on the table, along with an assortment of utensils that looked like they'd been borrowed from the Medieval Instruments of Torture exhibit at the London Dungeon. Together we'd carefully choose a recipe (already broken down into a helpful step-by-step guide), assemble the right ingredients, in the right amounts, and start our Sunday cook-along.

Mum was especially good at adding her own flair to recipes to make them her own. I think that's the difference between a cook and chef: a cook can follow a set of instructions and produce a pre-designed result, whilst a chef's experience and technical wizardry allows them to go off-piste to respond to their users – in this case, a hungry family of five – and adapt a recipe to their own needs. And it's the same for programming: starting to code for the first time can feel like you're following a bunch of steps in a guide without giving the process much thought. Over time, it becomes easier to introduce your own thoughts and ideas into your work, rather than just following instructions.

At the Raspberry Pi Foundation, we try to make coding less of a 'coding-by-numbers' experience for beginners by introducing them to project-based digital making activities from the start. By tackling real-world challenges like: 'how do I programme these LEDs in Scratch so that they run in the correct sequence for a traffic light system?' or 'how do I make this sparkly unicorn dance on my screen and my LED array?', they're making a project that's both fun and meaningful.

Running the program

After choosing the recipe, we needed to work out how to complete it in the correct sequence. In a recipe for walnut cake, for example, I needed to fold the walnuts into the cake mix near the end of the process so they weren't shredded in the blender in an earlier step. I learned that the order that operations were carried out was critical. Recipes are a great way to demonstrate how to solve a problem with a computing skill set. By taking an everyday food preparation task, for example, making a cup of tea (see my attempt in the box-out) and breaking it down into step-by-step instructions, you're developing your learner's computational thinking skills by teaching them sequencing, decomposition, and abstraction in a fun way. You can find out more about our thoughts on tea-making in our online training course on FutureLearn: Teaching Programming in Primary Schools.

It's also interesting to think about the roles that environment and experience play in how a learner might approach this task. Some might put the milk in first before pouring the water – the horror! – whilst others might add a blend of herbs or spices, in the traditions of their country. What's fascinating is that all these methods work: it would still be a recognisable and perfectly drinkable cup of tea, but the learners respond to the challenge in a way that's meaningful and personal to their own contexts. We should likewise encourage students to solve problems in their own way and not be afraid of experimenting with different solutions.

> “RECIPES ARE A GREAT WAY TO DEMONSTRATE HOW TO SOLVE A PROBLEM WITH A COMPUTING SKILL SET

Baking our creation

Back to our cake, though: a few soggy bottoms later and I learned that you need to create the right environment for a cake to rise properly. Not enough heat, and it came out anaemic and pale-looking. Too much heat, and it was burned to a crisp. It was also possible that the heat was just right, but I'd missed out a step in my sequence, or I'd baked it for too long. So many variables to get right in one small sequence: no wonder I felt such a sense of achievement when seven-year-old me produced a golden-brown sponge for the first time!

This got me thinking about the process of debugging and being resilient to setbacks. Just like my burnt cake that was salvaged by cutting the crispy bits off with a knife and covering it in icing, a learner's code is always only a couple of error messages away from working. They just need to breathe, evaluate what went wrong, make some changes, and keep persevering towards that sweet, sweet taste of success.

SEQUENCING A CUP OF TEA

- Put teabag in cup
- Switch on kettle
- While kettle is boiling, get milk out of fridge
- Wait until kettle has boiled
- Pour boiling water on teabag
- Wait two mins for brewing to occur
- Dunk teabag twice
- Remove teabag with spoon
- Pour [x]ml milk to taste
- Stir with spoon
- Remove spoon
- Enjoy!

User testing

The best part of the process is, of course, when you get to eat the cake. For me, there were a couple of times where this didn't go as planned. I remember one experiment where I boldly added some sardines to an Angel Food cake mix, which was met with my very first one star review as a cook. Sad times. But it does bring up the important point of user testing and how it's important to think about the experience of the end user for the thing you're making. Challenging students to discuss and think about how others might react to their project is a useful exercise and will hopefully encourage them to build more user-friendly interfaces in the future. This process of being playful, creative, and having the confidence to constantly improve a project through evaluation and iteration are all useful skills for the budding computer scientist.

What other fun, non-computing activities could you do with your learners as a gateway to getting them excited about Computer Science? (HW)

EVENTFUL INSPIRATION: PART 4

Our guide to running a successful tech-based event, packed with advice from people who are already running theirs

So you're thinking of running a tech-based community event? You want to get like-minded people together to encourage digital making, coding, and community? Well, this is your guide to planning and delivering your hackathon, techmeet or Jam.

WHAT IS A RASPBERRY JAM?

Raspberry Jams are independently organised community events where people get together to share knowledge, learn new things, and meet other Raspberry Pi enthusiasts. Jams provide opportunities for people to get involved in digital making, develop their abilities, get together, have fun, and socialise. They are usually free or very cheap to attend.

This series of articles is based on The Raspberry Pi Foundation's guide to running a community event. The Raspberry Jam Guidebook is aimed at Jam organisers, but the advice it offers will be help anyone setting out to run a tech-based community event.

Ben Nuttall, Community Manager at the Raspberry Pi Foundation, compiled the guidebook. He collected advice from existing Jam organisers throughout the UK, on everything from finding a venue to managing your finances, and from planning your activities to managing social media. Packed full of great, first-hand advice, the guidebook is designed to help you to run the best event you possibly can.

In issue 4 of Hello World, we considered health and safety, safeguarding, and money. In this issue, we look at planning the big day, and making your first event happen.

Volunteers

To run your event, you're going to need as many volunteers and helpers as you can find. For your first event, ask existing members of the Raspberry Pi community for help. You could also look for helpers in the wider digital making community, so that means making contact with Code Clubs, CoderDojos, hackerspaces, and other local tech groups.

Helpers are important. Workshop leaders will appreciate a second pair of hands during their activity, speakers will

OUR VOLUNTEERS HAVE BADGES THAT SAY 'JAM MAKER' ON THEM. SOMETIMES WE HAVE LANYARDS. TIM AND I TEND TO WEAR OUR PURPLE CAMJAM POLO SHIRTS. THEY WEREN'T CHEAP, BUT WE DO STAND OUT!

MICHAEL HORNE
CAMBRIDGE RASPBERRY JAM

WE HAVE A MAIN CHECK-IN DESK BY THE FRONT DOOR. AS FOLKS COME IN, THEY ARE CHECKED OFF THE EVENTBRITE LIST BY SOMEONE SEARCHING FOR NAMES ON ONE OF THE LAPTOPS, OR BY SCANNING THEIR TICKET VIA THE MOBILE APP. AS WELL AS THE MAIN EVENTBRITE ATTENDEE LIST, WE ALSO MAINTAIN A SEPARATE LIST AS PEOPLE ARRIVE, SO THAT WE KNOW WHO IS IN THE BUILDING IN CASE OF A FIRE.

ANDREW MULHOLLAND

NORTHERN IRELAND RASPBERRY JAM

need someone to introduce them and manage timings, and beginner sessions need people to guide newcomers through the activities. If you're checking people into the event, you'll need volunteers helping on the door. The more volunteers you have, the easier everything will be, and the less each person will have to do.

It's good practice to waive the ticket fee for helpers. Make sure your helpers know what their roles are, and that they're comfortable with their tasks. Make sure your helpers feel appreciated and valued, and that they're able to enjoy the event as well as taking part.

Make it obvious who the helpers are, and make sure attendees know who they can go to for help. Volunteer name badges, lanyards or T-shirts are useful for identifying helpers and making it easy for attendees to approach them. They'll also help your volunteers to feel part of your team.

WE USE THE EVENTBRITE APP TO CHECK IN ATTENDEES WHERE POSSIBLE. HOWEVER, PEOPLE TEND TO ARRIVE ALL AT ONCE AND WE DON'T LIKE A QUEUE. IF IT'S NOT POSSIBLE TO SCAN THE TICKETS, WE JUST CHECK THEM AS THEY COME IN. IT MEANS OUR NUMBERS ARE A LITTLE OFF, BUT IT KEEPS PEOPLE HAPPIER.

MICHAEL HORNE

CAMBRIDGE RASPBERRY JAM

Schedule

If your Jam features scheduled activities, draw up a timetable for the day. If you can, share it with attendees before the event. Display your schedule in the venue on the day, on screens, posters, or handouts, to help people to plan their activities.

Checking in

If you plan to check your attendees into the event, print out a list from your Eventbrite page, and ask a volunteer to tick people off the list as they arrive. You can also check people in on a laptop using the Eventbrite website, or on a tablet or phone using the Eventbrite app.

Give out name badges to attendees (a sticky label will do). This will make them feel more welcome, and encourage them to socialise with other people at the event.

WE'RE NOT ALLOWED FOOD AND DRINK INSIDE OUR VENUE. THE EVENT STARTS AT 1PM SO WE ADVISE PEOPLE TO EAT BEFORE THEY ARRIVE, AND WE POINT OUT SOME LOCAL CAFÉ.

ANDREW OAKLEY

COTSWOLD JAM

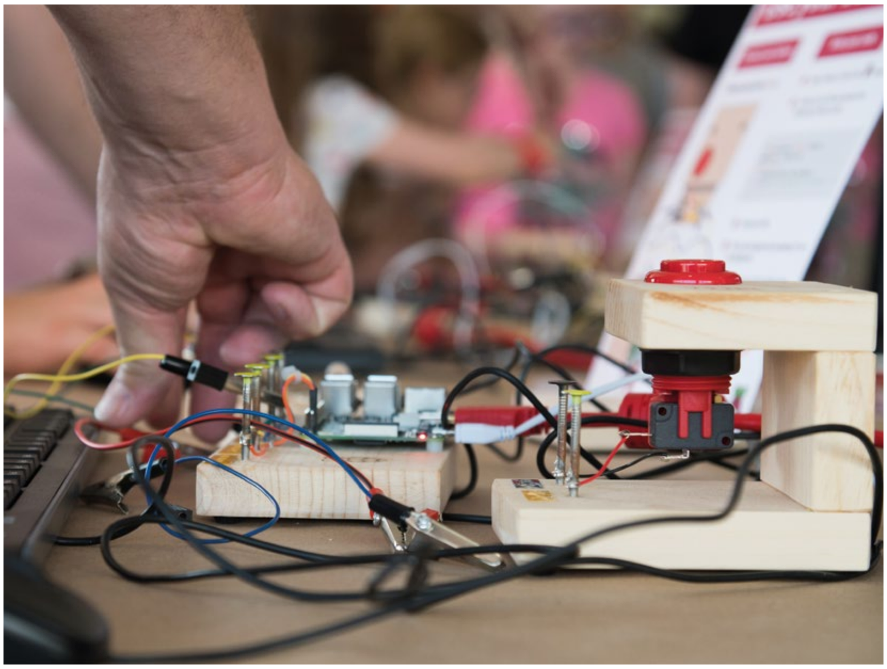
If you're taking photos for social media, give your attendees (especially parents) photo release forms to sign when they check in.

These forms will grant you permission to use photos of your attendees online, or for other publicity. If anyone refuses permission, let your photographers know. For example, giving these people a name badge in a different colour will help the photographers to spot people who don't want to be in photos – both

WE SEARCHED ONLINE FOR CATERERS IN A 30-MILE RADIUS OF THE CAMJAM VENUE, AND MANAGED TO FIND SOMEONE WHO WOULD ATTEND THE EVENT FREE OF CHARGE. FOR PI WARS, AT THE CAMBRIDGE COMPUTER LAB, WE ARE OBLIGED TO USE THEIR INTERNAL CATERING WHICH COSTS US £200 PER DAY TO COVER STAFFING COSTS. IT'S VERY CONVENIENT, FOR US AND THE ATTENDEES, AND WE KNOW WE CAN BUILD THAT INTO THE TICKET PRICE.

MICHAEL HORNE

CAMBRIDGE RASPBERRY JAM



WHY RUN YOUR OWN EVENT?

We asked people running Raspberry Jams what inspired them to start their own tech-based community event.

I GOT ONE OF THE FIRST RASPBERRY PIS AND WAS LOOKING TO FIND OTHER PEOPLE WHO HAD LITTLE EXPERIENCE OF PROGRAMMING AND LINUX TO HELP AND SUPPORT EACH OTHER. I APPROACHED THE LIBRARY TO SEE IF THEY'D GIVE US THE SPACE AND THEY AGREED.

SIMON BELSHAW
EXETER RASPBERRY JAM

I WANT TO BE ABLE TO GIVE SOMETHING BACK TO THE COMMUNITY. I AM PASSIONATE ABOUT TEACHING KIDS TO PROGRAM AND INTERFACE WITH THE PHYSICAL WORLD.

STEVE AMOR
CORNWALL TECH JAM

I GOT INVOLVED BY BEING INTERESTED IN TECHNOLOGY, WHICH THEN GREW INTO A LOVE FOR RASPBERRY PI. I FOUND THERE WASN'T ENOUGH OF A RASPBERRY PI COMMUNITY WITHIN MY AREA OF SCOTLAND SO I WANTED TO DO SOMETHING ABOUT IT.

KERRY KIDD
DUNDEE RASPBERRY JAM

I WENT TO PICADEMY, AND THAT GAVE ME THE CONFIDENCE IN USING THE TECHNOLOGY TO ORGANISE MY FIRST JAM.

KATHARINE CHILDS
NOTTINGHAM RASPBERRY JAM

I WANTED TO KEEP UP TO DATE FOR THE CHILDREN IN THE CODE CLUB I RUN. I LEARN SO MUCH, AND I ALWAYS COME BACK INSPIRED WITH NEW IDEAS FOR CODE CLUB.

ANNE CARLILL
YORK RASPBERRY JAM

I LIKE TO GIVE SOMETHING BACK TO THE COMMUNITY, AND WORKING WITH THE KIDS WHO COME TO THE JAM IS A REWARDING CHALLENGE THAT KEEPS MY SKILLS SHARP.

LES POUNDER
BLACKPOOL RASPBERRY JAM

I LOVE TO LEARN. BY BRINGING MAKERS AND OTHERS TOGETHER, I GET TO SEE A VARIETY OF THINGS AND DISCUSS AMAZING IDEAS. IT'S GREAT TO SEE EIGHT-YEAR-OLDS AND EIGHTY-YEAR-OLDS MAKING LIGHTS BLINK AND DINOSAURS NOD.

LUCY ROGERS BLACKGANG CHINE

AFTER HAVING SUCH A GREAT TIME AT CAMJAM AND OTHER EVENTS, I DECIDED TO START MY OWN IN BIRMINGHAM.

SPENCER ORGAN

BIRMINGHAM LEARNING HUB

I'VE MADE SOME GREAT FRIENDSHIPS BECAUSE OF THE JAMS. CONTINUING TO ORGANISE THEM MEANS I GET TO CONTINUE WITH, AND TO BUILD ON, THOSE FRIENDSHIPS.

MICHAEL HORNE CAMJAM

at the event, and when they're reviewing their images later.

Catering

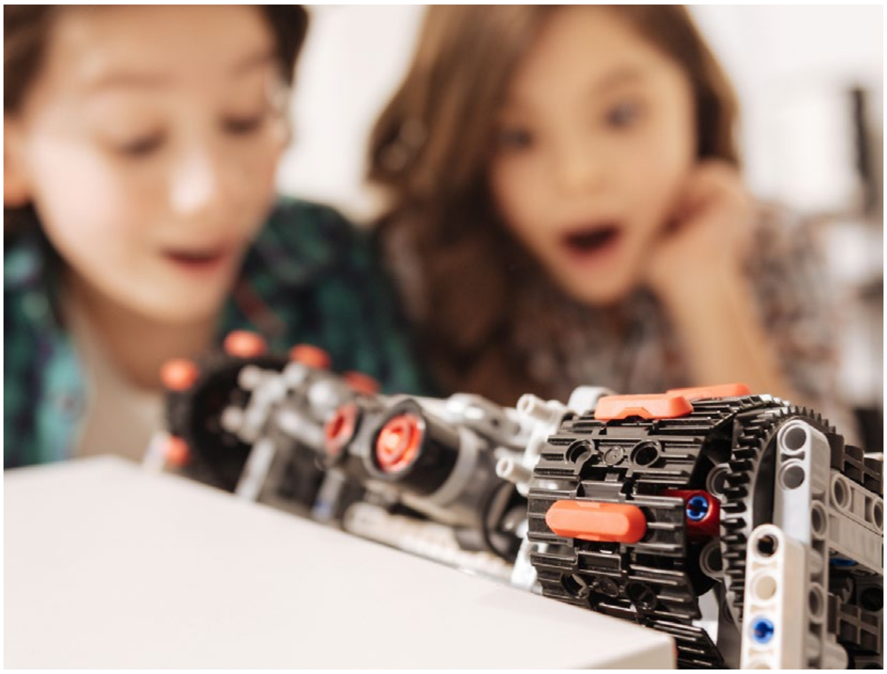
Some Jams use part of the ticket price to provide catering. Others collect donations and order pizza on the day. If your venue has a café, make sure it's open and accessible to your attendees during your event. Access to water, tea, coffee, juice, and biscuits will keep most people happy.

Whether or not you provide catering, your attendees will need access to food and drink. Provide information on nearby shops and cafés, and set up a re-entry system to allow people to go out for refreshments. If your venue is on a high street, or close to plenty of shops and cafés, it might be easier to ask people to source their own food.

If you're providing food, make sure you have an allergy policy. You can't assume that everyone can eat what you provide, and it might not be obvious whether allergens are present. This uncertainty could mean that some attendees are unable to eat, or it could lead to a serious allergic reaction. Ask people to provide details of their allergies and other dietary requirements in advance, and think about how you're going to cater for them.

Reflection

After your event, take some time to reflect on how everything went. Ask yourself:



> IF YOUR VENUE HAS A CAFÉ, MAKE SURE IT'S OPEN AND ACCESSIBLE TO YOUR ATTENDEES DURING YOUR EVENT

- Was it a good event?
- What did people like about it?
- Would you do it again?
- If you were to run another event, what would you do differently?
- How often will you run the event in the future?

You might be ready to start planning your next event straight away. That's great! If you can use the same venue again, choose another date, set up an Eventbrite page, and start promoting the next event. If running the first event was an overwhelming experience, take a break and don't rush into the next event. Think about what you would do differently. Organise more support for your next event. For help with your next steps, reach out to members of the Raspberry Jam community, and other local tech-based organisations. (HW)

RASPBERRY
JAM
GUIDEBOOK
STARTING AND RUNNING A RASPBERRY JAM
ADVICE GATHERED FROM THE RASPBERRY PI COMMUNITY

DOWNLOAD

Download your copy of the Raspberry Jam Guidebook from: **helloworld.cc/2p7ZX98**

REFLECTION

Flexible formats

When I ran my first Jam, I thought of it as being a meetup for techie adults, but I decided to run the first one on a Saturday. When kids and families turned up, I realised this was a better format than I had had in mind. It was great to see people of all ages enthused by technology, so we kept the free-form practical format, and it worked really well.

BEN NUTTALL
MANCHESTER RASPBERRY JAM

Planning ahead

I know I'll need some good ideas to move beginners on, or they won't return. I know I will have to advertise more widely to get the same number of people next time. People always come in greater numbers to the first of something new.

ANNE CARLILL
YORK RASPBERRY JAM

Dream big

To begin with, we aimed at introducing coding and physical computing to young people, parents, and teachers through Raspberry Pi. Now we are developing the digital making skills of our regular young coders through collaboration on projects like Pioneers, entering competitions like Astro Pi, and developing young leaders who are going to be teaching others. We want to link up with other young leaders in other parts of the country too. We are even planning a road trip to Bangladesh to share experiences with other young people. It's all very exciting!

GRACE OWOLADE-COOMBES
SOUTH LONDON RASPBERRY JAM

NO NEED TO PINKIFY TECHNOLOGY SUBJECTS FOR GIRLS

It's time to avoid using female-friendly stereotypes in order to encourage girls into enjoying ICT or computing concepts...

STORY BY Stephen Trask

Computer science within UK schools is in an interesting position right now. In addition to the debate about its suitability for the wide interest and ability ranges across UK students is the observation that girls are woefully under-represented. The statistics are damning – Computing and ICT-related GCSEs are taken by only 1.5% of female students, compared to 3.7% for males. Only 9.8% of students pursuing a Computing A Level course were girls last year.

There have been attempts to redress this imbalance with a 'pinkifying' approach to STEM subjects, to make something such as coding more attractive to girls. In a nutshell, this means printing documents on pink and purple polka-dot paper. Creating fonts that are pink and flowery. Purple websites.

As the father and teacher of three primary school children, two of whom are girls, I have the weekly pleasure of teaching my daughters to code. I've taught in all-boys schools, all-girls schools, co-ed schools, and international schools. The subjects have ranged from Information Technology, ICT and Computing. Yet at no point did it occur to me that pushing pink, girly stereotypes was essential to recruit more girls into my subject. It is true that boys tend to pick such subjects more than girls, and there are a whole host of reasons why this might be, including gender stereotypes from society at large. However, to combat the stereotype that boys are just into tech with the stereotype that girls just want flowery pink items is to completely miss the point.

Girls respond to positive role models within society and within schools, to challenging concepts, to having their eyes opened to the sort of STEM concepts that can spring from a KS2 or KS3 lesson, such as a discussion about cybersecurity or AI in self-driving cars. Funnily enough, so do boys. I teach a wide age range, so I experience what it's like to teach children in their formative years, as well as when they're figuring out what adolescence means. If you pop into one of my classes, you'll see girls coding HTML and CSS in Year 4. You'll see them utilising the principles of BIDMAS in Python programming in Year 9. You'll see a lively Google Classroom debate about the ethics of self-driving AI entities in military combat situations. It didn't occur to me to encourage the girls to create a website about My Little Pony to hook them into my subject, any more than I'd encourage the boys to focus more on Star Wars or Fortnite.

My daughter in Year 4 is obsessed with puppies, but the website she created recently was about *The Clone Wars* cartoon series. My son in Year 6 still happily watches *Waffle the Dog* with his younger sisters. His recent Scratch game involved a boss-level character of a pony with love-heart laser beam eyes as a weapon. One of my star Year 9 students has recently been coming into my computer lab at lunchtime to work on her Python adventure game. In the majority of scenes, you meet a grisly end at the hands of space monsters and haunted beings. Not a hint of pink.

If I was to heavily encourage the females in my class to use more 'girly' concepts or topics, what does that communicate? That gender stereotypes are fine as long as girls take up my subject at GCSE? That boys should avoid female stereotypes, because there's no place on an Engineering course at university for boys who code ponies with laser beam eyes? Why would I want to perpetuate such ridiculous myths in an age where many schools are seeking to be more inclusive?

All colours and gender preferences should be welcomed. To that end, I try not to introduce male or female stereotyped concepts if I'm setting a project. Only a couple of years ago, IBM put out a call to women in science and technology to 'hack a hairdryer'. The aim was to encourage young females into tech. Because clearly hacking a hairdryer would be more interesting and, er, relevant to them than something more boyish. Such as, um, car engines or space flight. IBM is the company behind Watson, and I wish they'd consulted their own vast AI technology to see what Watson would have recommended. Equally, if girls decide that they want to write pseudocode for baking something quite complicated, because they like *The Great British Bake Off*, why should that be a problem? It should be no more of a problem than if boys take the same topic. As educators, we should avoid pushing one activity to one particular gender because we think it will hook them, when all it might do is reinforce the old stereotypes that got us here in the first place.

My aim is to guide the children I teach and father towards a world that's free from gender stereotypes because they are the generation that will have the power to change these outmoded concepts. (HW)

Stephen has worked in a variety of technology-related senior leadership roles in places as diverse as Singapore, Kuala Lumpur, Abu Dhabi and West Sussex. He's currently overseeing technology and data in the New School, Rome, Italy, and he teaches Computing from Year 1 to Year 11.

MILES BERRY PRINCIPAL LECTURER

TEACHERS' KNOWLEDGE

Teaching as an engineering discipline

We talk of the 'art of teaching': that teaching is akin to performance, an inherently creative process, and that engagement is key to its success. We talk also of the 'science of teaching': that our approach should be informed by research, that results should be measurable and that data is central to the process. Teaching is both an art and a science, but for the pragmatic educator, perhaps it's helpful to think of teaching as closer to an engineering discipline.

Teaching as engineering would acknowledge that what we do as educators is creative, in the literal sense of making things: the learning environment, resources for our learners (always one of my favourite parts of the job), and learning experiences (whether recorded in lesson plans or not). More importantly, as educators we make connections inside our learners' heads: connecting neurons which weren't connected before, or strengthening the connections that are already there. Unless our learners' brains have changed, can we really say we've taught anything?

Making progress

Teaching as engineering also acknowledges the fundamental moral purpose to what we do: were it merely an art, it would be sufficient to engage and entertain; were it just science, it would be enough to understand how our students learn and how we might most effectively teach; as an engineering discipline, our focus has to be on ensuring our students make progress, that they can do more, know more and understand more than they would otherwise. We share with engineers the core engineering mind of 'making things that work and making things work better', even if the 'things' here are less tangible than a bridge, a circuit board or a software system.

Just as engineering is the systematic application of science, so education is, or should be, the systematic application of what we know about how learning happens to the acquisition of new skills, knowledge and understanding by our students. This is why it matters that educators understand enough of how the brain works, and how a group of learners engage and interact, to be able to apply this knowledge to creating the resources, experiences and environment through which learning will happen.

Problem finding

Jon Chippindall's discussion of 'engineering habits of mind' (page 22) is as relevant to education as it is to other engineering disciplines: when we teach new content, or work with new classes, we're problem finding; the best educators adapt and apply the resources and plans of others; systematic, holistic thinking is essential for the educator; educators should be relentlessly seeking to improve their practice and their students' outcomes; visualisation gives us, and our students, a way to think about complex ideas and systems; and the job of a teacher involves creative problem solving on a lesson-by-lesson, day-by-day and year-by-year basis.

Engineering wouldn't be possible without engineering education: few of us in schools have a background in engineering, but if we think of teaching itself as a sort of engineering then all of us can perhaps help cultivate this way of looking at the world in the young people with whom we work. (HW)

Miles Berry is principal lecturer in computing education at the University of Roehampton. He is a member of the CAS and CSTA boards and the Raspberry Pi Foundation.

"HELLO, WORLD!"

Everything you need to know about the new computing and digital making magazine for educators

Q WHAT IS HELLO WORLD?

A Hello World magazine is a new magazine for computing and digital making educators. Written by educators, for educators, the magazine is designed as a platform to help you find inspiration, share experiences, and learn from each other.

Q WHO MAKES HELLO WORLD?

A The magazine is a joint collaboration between its publisher, Raspberry Pi, and Computing At School (part of BCS, The Chartered Institute for IT). Hello World is sponsored by BT.

Q WHY DID WE MAKE IT?

A There's growing momentum behind the idea of putting computing and digital making at the heart of modern education, and we feel there's a need to do more to connect with and support educators inside and outside the classroom.

Q WHEN IS IT AVAILABLE?

A Your new 100-page magazine will be available three times per year in time for each new term in January, April, and September. Would you like it to be available more frequently? Let us know!

THE MAGAZINE FOR COMPUTING & DIGITAL MAKING EDUCATORS
(Hello World)
Issue 5 Summer Term 2018 helloworld.cc
DATABASES AND BIG DATA
Why do many struggle with databases - and what can you do about it?
INSPIRING ENGINEERS!
Ideas and inspiration to get your students thinking like engineers
Coding and INCLUSION
Helping visually impaired students realise their potential
ORGANISING A COMPUTING DAY OUT
Where to go, and what to do when you get there...
PSSST! IS IT OKAY TO COPY?
Why building on existing code is great for teaching
PLUS FLOW CONTROL IN PRIMARY PROGRAMMING • HOW WE BUILT THE RASPBERRY PI • WHY IT'S WORTH TRYING GP! AFTER SCRATCH • MICRO:BIT • TANGIBLE PROGRAMMING IDEAS • 2018: YEAR OF ENGINEERING

IT'S FREE!

Hello World is free now and forever as a Creative Commons PDF download. You can download every issue from **helloworld.cc**. Visit the site to see if you're entitled to a free print edition, too.

WANT TO GET INVOLVED?

There are numerous ways for you to get involved with the magazine.
Here are just a handful of ideas to get you started:

- **Give us feedback**
  Help us make your magazine better – your feedback is greatly appreciated.
- **Ask us a question**
  Do you have a question for a FAQ section or a bugbear you'd like to share? We'll feature your thoughts and ideas.
- **Tell us your story**
  Have you had a recent success (or failure) you think the wider community would benefit from hearing? We'd like to share it.
- **Write for the magazine**
  Do you have an interesting article idea? We'd love to hear from you.

GET IN TOUCH

Want to talk? You can reach us at:
contact@helloworld.cc

FIND US ONLINE

www.helloworld.cc

@HelloWorld_Edu

fb.com/HelloWorldEduMag

SUBSCRIBE IN PRINT TODAY!
PAGES 28-29

(Hello World)

helloworld.cc