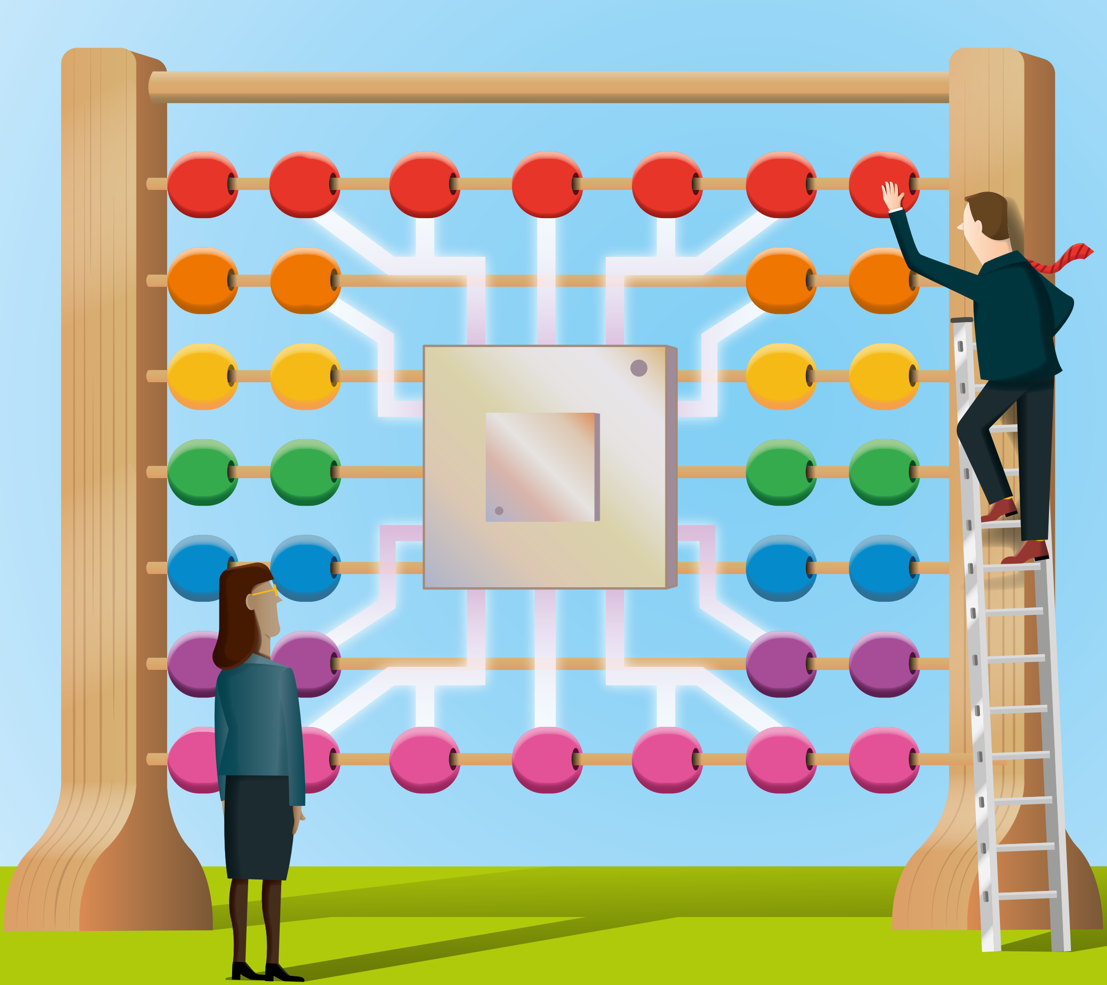
THE MAGAZINE FOR COMPUTING & DIGITAL MAKING EDUCATORS

(Hello World)

Issue 10 | Oct 2019 helloworld.cc

WHY TEACH CHILDREN TO CODE?

The role of programming in CS



CYBERSECURITY IN CLASS

Where to get started

PLANNING THE PERFECT OPEN EVENING

Insider's guide inside

SORTING OUT SHAPES

Logo lesson tips

LET IT GROW!

When technology meets the school garden

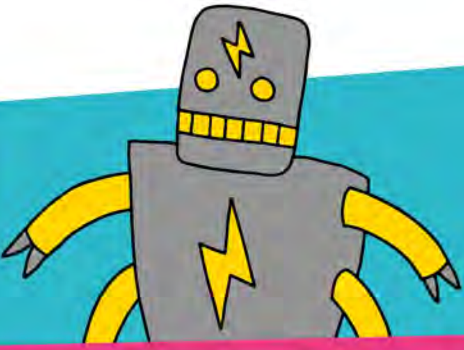
SEMANTIC WAVES

Ideas for reviewing lesson activities

MATHS & COMPUTER SCIENCE

Fresh ideas and activities for your classroom

PLUS BRINGING PHYSICAL COMPUTING TO THE CLASSROOM • MAKING NEWS AT SKY • CODE CLUBS FOR TEACHERS RETRIEVAL PRACTICE & LEARNING • GOING BEYOND THE QUESTION • IMAGINING THROUGH VIRTUAL REALITY



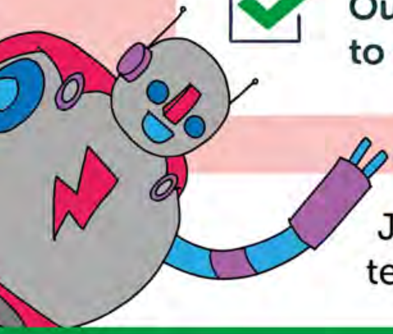
START A CODE CLUB IN YOUR SCHOOL!

It's easy to get your school coding! Code Club supports educators to run coding clubs for 9- to 13-year-olds with free, step-by-step project guides for learning Scratch, Python, and HTML/CSS.

"I run a Code Club in our school for children in Year 5 and 6, which is always popular. Code Club activities encourage learning independence, and the children love to share the games they've coded with each other."

Jill, Teacher

- ☑ **Code Club is free**
- ☑ **Code Club offers engaging, hands-on activities that have been designed to spark curiosity and inspire creative thinking**
- ☑ **Our free online training is perfect for teachers who are new to coding or are starting a Code Club for the first time**



Join the network of over 7000 clubs across the UK teaching more than 100,000 young people to code!

Get involved at codeclub.org

Code Club is part of the Raspberry Pi Foundation (registered charity no. 1129409)

THE MAGAZINE FOR COMPUTING & DIGITAL MAKING EDUCATORS
(Hello World)
WHY TEACH CHILDREN TO CODE?
CYBERSECURITY IN CLASS
PLANNING THE PERFECT OPEN EVENING
SORTING OUT SHAPES
MATHS & COMPUTER SCIENCE
PLUS

SUBSCRIBE IN PRINT FOR FREE
TURN TO PAGE 28

HELLO, WORLD!

Mathematics. A subject I've had a love-hate relationship throughout my life. I love logic, patterns, programming, and computer architecture, which can all be linked back to mathematical thinking. Computer Science is still defining itself in relation to other disciplines like engineering, science, and maths, so in this issue of *Hello World*, we're taking a look at computing through a mathematical lens.

Michelle Ainley shares her experience of using *ScratchMaths* to develop her students' confidence in computational thinking, as well as developing her own practice. Whilst **Laura Sach** explores how sometimes mathematical terminology can obscure otherwise straightforward concepts which lead to misconceptions and frustration.

You can count on us to bring you the latest research and classroom practice as always in this edition, with articles too on using technology to monitor a school garden (and make sure its plants are okay), designing with virtual reality, and using more free-free approaches in the classroom. Don't miss too our extensive guide to running an open day for prospective new students: we really are back at that time of the year already.

We are passionate about sharing your experiences of teaching computing and digital making with educators all over the world, please send an email to **contact@helloworld.cc** if you'd like to write something for a future edition.

Carrie Anne Philbin
Raspberry Pi Foundation

FEATURED THIS ISSUE

RUTH WILLENBORG

Ruth is a retired IBM Distinguished Engineer, is the founder and champion of IBM CoderDojo Research Triangle Park (RTP), NC and a Triangle Techgirlz volunteer instructor. She writes about her Scratch challenges and more on page 42.

CHRIS AVILES

Chris is a Raspberry Pi Certified Educator and teacher at Knollwood middle school in the Fair Haven school district in Fair Haven, New Jersey. He tells us about his school garden – with a technology twist – on page 30.

MICHELLE AINLEY

Michelle is a computing teacher at Naima Jewish Preparatory School in West London. She is also involved in CAS Westminster. She takes us hands-on with the teaching of ScratchMaths as a non-specialist over on page 19.

(HW)

EDITORIAL

Publishing Director
Russell Barnes
russell@raspberrypi.org

Editor
Sian Williams Page
sian@helloworld.cc

Contributing Editors
Miles Berry
miles@helloworld.cc

Simon Brew
simon.brew@raspberrypi.org

Sub Editors
Vel Ilic, Nicola King, Phil King

Subscriptions
Joshua Crossman

DESIGN

Critical Media
criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Harriet Knight, Sam Ribbits, Ty Logan

Photography:
Brian O'Halloran and Fiacre Muller / Raspberry Pi Foundation
Adobe Stock Photo

Cover illustration
Sam Alder

Graphics
Jack Willis

CONTRIBUTORS

Michelle Ainley, Chris Aviles, Miles Berry, Mark Calleja, Gareth Edgell, Kelly Frost, Sway Grantham, Will Grey, Dr. Andreas Grillenberger, Amanda Haughs, Pen Holland, Amanda Hubball, Andreas Koch, Thom Kunkeler, Dr Ursula Martin, Karl Maton, Greg Michaelson, Gemma Moine, Di Norvock, Alan O'Donohue, Spencer Organ, Alex Parry, Simon Peyton Jones, Nicholas Provenzano, Mareen Pryzbyalla, Oliver Quinlan, Chris Roffey, Laura Sach, Amy Stancer, Lucinda Tuttiett, Jane Waite, Ruth Willenborg, Matthew Wimpenny-Smith, Sarah Wyse

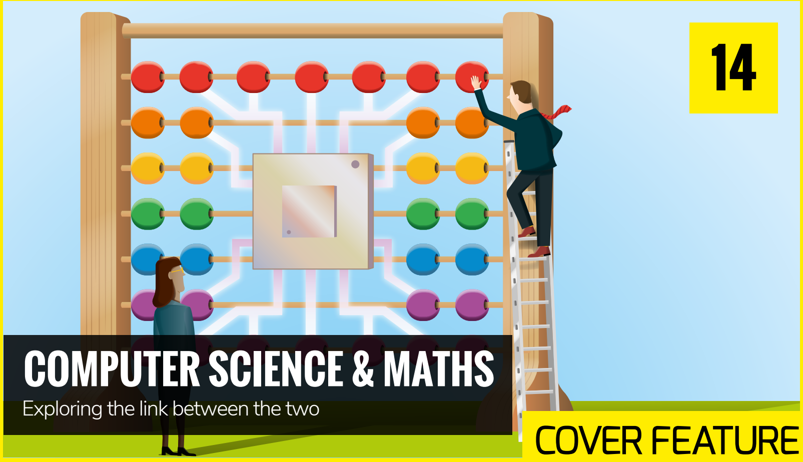
Hello World is a joint collaboration:

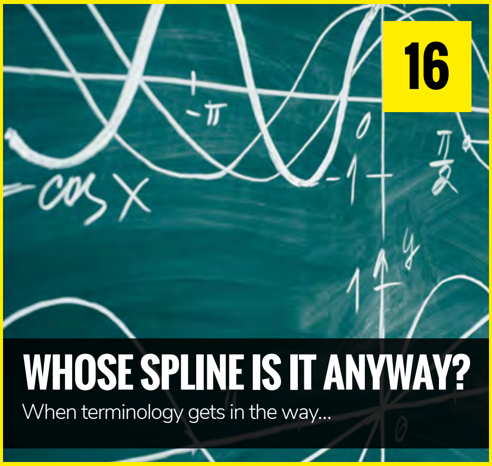
Raspberry Pi
bcs The Chartered Institute for IT Academy of Computing
CAS

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., Maurice Wilkes Building, St. John's Innovation Park, Cowley Road, Cambridge, CB4 0DS. 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





NEWS, FEATURES, AND OPINION

CONVERSATION

REVIEWS

LEARNING

RESOURCES & LESSON PLANS

FREE NEW RESOURCES FROM NCCE

The first units of work are launching as part of the National Centre for Computing Education's resource repository, and you can be some of the first educators to pilot them

Carrie Anne Philbin

Work has been under way at the National Centre on a comprehensive collection of over 500 hours of teaching materials, facilitating the delivery of the entire computing curriculum from Key Stages 1 to 4 in England. The aim is to help reduce teacher workload whilst increasing subject knowledge and a greater understanding of effective pedagogy. All resource repository content will be free, open, and editable (under the Open Government Licence (OGL)), ensuring the resources can be tailored to each individual teacher and school setting, and that they are suitable for all students, regardless of their ability, background, and additional needs.

On 30 September 2019, the first units developed by educators will be published on **teachcomputing.org**. Over the next nine months, this collection will grow to approximately 36 units for primary computing and 30 for secondary computing, including GCSE Computer Science.

What the resource repository will include

The resource repository, once completed next year, will encompass the following:

- A teacher manual for each key stage, which will include support materials such as aspirational pedagogy and best practice, along with guidance on inclusion and assessment.
- At least six units of work per year group, including an overview of a connected series of lessons, detailing context, background, and student progression. Each overview will also link to professional development opportunities offered by the National Centre to assist teachers in developing their subject knowledge.
- Approximately six lesson plans per unit of work. These contain all the information required for a teacher to deliver a particular lesson, including any prior reading, plus details of equipment and resources required to deliver particular activity to students of varied abilities.
- Homework and assessment for each unit of work from Key Stage 2 onwards to be completed either to extend or assess students' learning.
- A number of activities for each lesson that form a discrete 'chunk' of learning. These may be online or offline, and take many varied forms. Some of these activities include suggested adaptations to suit a variety of students and their needs.
- A learning graph – a visual representation of a particular topic, including the relationships and dependencies between skills and concepts.

HOW TO ACCESS THE RESOURCE REPOSITORY

To ensure easy access, the content will be discoverable by simply logging into **teachcomputing.org** and selecting an academic year group within a Key Stage. You can browse any of the content and share it, as well as make your own copy to adapt and print.

In the future, mechanisms will exist for you to provide a rating and detailed feedback to help make the materials as useful as possible.

What about other resources?

There are many fantastic resources produced for teaching computing in England, like Barefoot Computing and Rising Stars, as well as outstanding lessons and activities contributed by educators to CAS resources.

The aim of the National Centre's resources is to provide a comprehensive learning pathway for computing education, from Year 1, through transition to secondary, resulting in a qualification in Year 11. In the next few years, the repository will grow to include links to high-quality published resources. Furthermore, by providing easy access to good content, teachers are able to use their time more effectively to focus on enhancing their subject knowledge and pedagogical practice.

Tried and tested?

One of the reasons that it takes a long time to develop all of the content to teach computing is because it is piloted in schools. During the last term of the 2018/19 academic year, our first units of work were piloted in primary and secondary schools in England. During the pilots, we asked both specialist and non-specialist teachers of computing to deliver content-related to computing systems and networks to children in Years 1 to 3 in primary and data representation to Year 8. Some provided feedback after the lessons took place, whilst others allowed our content writers to sit in on the lessons to observe first-hand how the learners responded.

We've learned a great deal from these pilots, as well as from feedback from academics and specialist teachers. For example, as a result of the feedback, we will ensure a focus on providing clear, age-appropriate definitions of key vocabulary for children, outline further approaches to differentiation, and include additional assessment opportunities.

Overall, we have been delighted with the pilots – particularly the comments from non-specialist teachers, who found that the lesson plans enabled them to deliver sessions effectively. One Year 3 teacher wrote:

"The lesson plan was very clear and easy to follow, with added explanations (which I feel I needed) to help with subject knowledge."

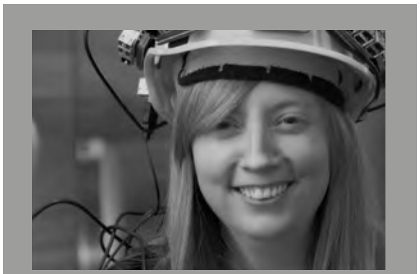
We were also very pleased to hear that teachers felt our activities were accessible to children, which enabled them to meet the lesson objectives:

"All the children in my class were able to access the resources for this lesson. [...] The final activity helped show if they understood what they had learnt / met their objective."

Register your interest in piloting resource repository content

As the next set of resources are developed, we're looking for more schools to be involved in piloting content. Please register your interest at **ncce.io/resourcespilot**. Give as much detail in the form as you can.

By participating in this pilot, you will help us provide better computing and computer science education support for all teachers in England and beyond. (HW)



CARRIE ANNE PHILBIN
Director of Educator Support at Raspberry Pi Foundation, CAS board member, chair of CAS #include, author, and YouTuber.

STORYTELLING WITH CODE CLUB AND SHAUN THE SHEEP

Right now, Code Club is running an exclusive, global competition in collaboration with Aardman Animations around the new Shaun the Sheep Movie: Farmageddon

Janina Ander

Code Club is a worldwide network of free teacher- and volunteer-led coding clubs for young people, and it's part of the Raspberry Pi family. The Code Club team is currently collaborating with Aardman Animations, the studio behind the Shaun the Sheep movies, to bring an exclusive coding competition to more than 13,000 clubs around the world. In celebration of Aardman Animations' release of their brand-new *A Shaun the Sheep Movie: Farmageddon*, Code Club members are getting the chance to use Scratch to tell a story about Shaun and his new friend LU-LA, and to maybe win a cool prize for their digital creation.

To enter the Code Club 'Shaun the Sheep: Mission to Space' Competition, young learners create a Scratch animation based on a ready-made project guide provided by Code Club, and images provided by Aardman Animations.

In this Code Club competition, children use Scratch to tell a story about Shaun and his new friend LU-LA

LU-LA and Shaun in a still from their incoming film

> CODE CLUB MEMBERS ARE GETTING THE CHANCE TO USE SCRATCH TO TELL A STORY ABOUT SHAUN

KEY COMPETITION DETAILS

- Open to all registered Code Clubs worldwide
- About combining coding skills with artistic creativity and storytelling
- Based on the new film, *Shaun the Sheep Movie: Farmageddon*
- Led by a step-by-step Scratch project guide, available in multiple languages
- Suitable for beginners and advanced Scratch coders
- Creating an entry will take 1-2 Code Club sessions
- Closing date: 25 October 2019
- Find out more at **codeclub.org**

From beginner to advanced

The competition is accessible to all learners, regardless of their coding knowledge. Code Club's special 'Shaun the Sheep: Mission to Space' project guide is a new version of the second project in the Code Club Scratch curriculum, teaching learners to create an animation by using repetition to move sprites around the screen. It's a suitable project for beginners, and more advanced learners can easily expand on it to challenge themselves.

In addition, participants in this Code Club competition will be judged in one of two age categories: 11 and under, or 12 and above. Thus, the judges can fairly compare entries of learners with similar literacy and numeracy skills.

To help learners participate who aren't fluent in English, the Code Club team has leveraged the power of the Raspberry Pi translation community to make the 'Shaun the Sheep: Mission to Space' project guide available in multiple languages.

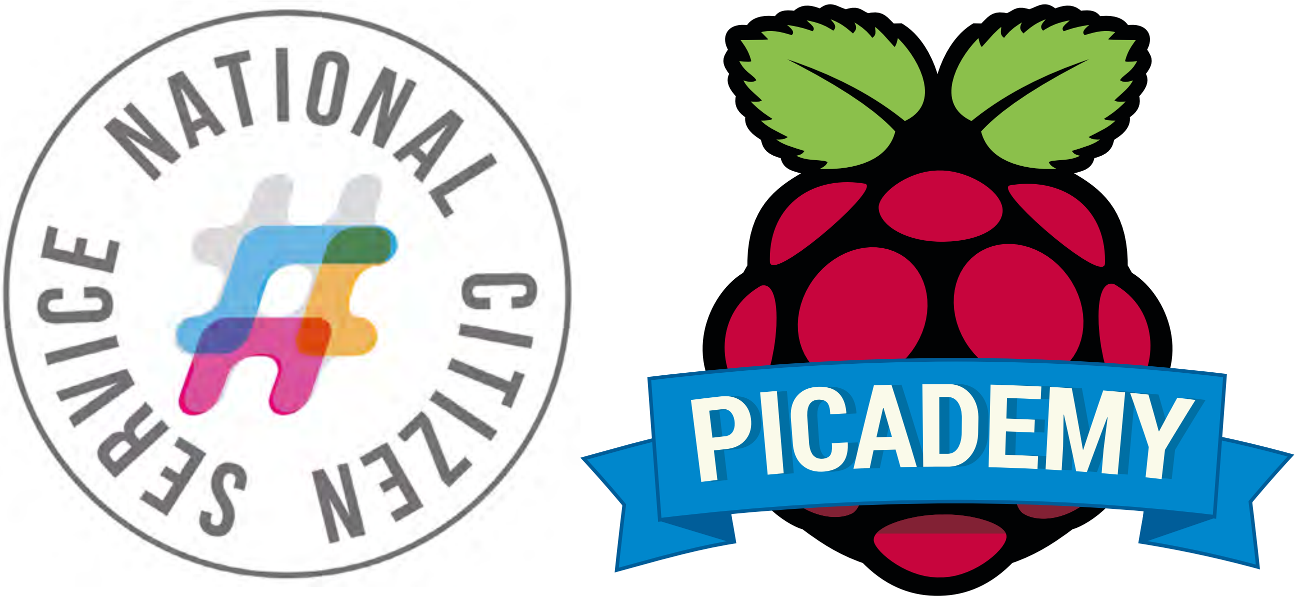
Promoting creative coding

Competition entries will not solely be judged according to the level of coding skills involved, but also according to how creative they are and how well they tell a story. This is because Code Club promotes coding as a creative endeavour. In Code Club sessions, young people get to progressively learn programming concepts while using code to create projects that reflect their interests and how they see the world.

The 'Shaun the Sheep: Mission to Space' Competition is exclusively open to registered Code Clubs, so if you're thinking about starting a Code Club, now is the time: this competition is perfect for getting your young learners motivated to try their hand at coding for the first time.

Head to **codeclub.org** now in order to get started. (HW)

A Shaun the Sheep MOVIE FARMAGEDDON™



SOCIAL ACTION HACKING:
TEACHING DIGITAL MAKING WITH THE NCS

Inspiring young people to create with technology requires a good reason for them to engage – the experimental new *Social Action Hackathon* programme has been created to do just that

Mark ‘Mr.C’ Calleja

For the third summer in a row, Raspberry Pi has been working alongside the National Citizen Service to present thousands of engaged young people with the opportunity to create something cool with technology for a worthy cause. The core principles of the NCS are to provide young people participating in its summer programme with important life skills for their futures, while exposing them to others of their own age from different viewpoints, classes, and cultural backgrounds.

It aims to provide opportunities for young people aged 15-17 to step up and join the innovators changing the world through social action and engagement. They’ve partnered with the Raspberry Pi Foundation to provide an authentic, relevant learning experience for young people, all in contexts and company they would not regularly experience in their daily lives.

The Raspberry Pi Social Action Hackathon programme was designed from the start to provide participants with an authentic and realistic experience of product development, from the point of view of a fledgling technology start-up. During the two-day Hackathon, young people engage with and participate in real-world development processes, modelled after the ‘Agile’ methodologies used every day in the technology industry.

Participants engage entirely in learner-led, project-based learning from the very outset of the course. They are split into small groups (which we refer to as ‘start-ups’) and are led through discussions and workshops. These provide some context for thinking about how technology can impact and improve the lives of people generally, and how this can apply to both charity beneficiaries and the providers of support to the disadvantaged within the UK.

Playing the game

In 90-minute workshops over their first day, participants are shown the capability and extent of the technology available to them. They undertake short, confidence-boosting programming activities to get them thinking about what assistive technologies they could create with the resources available. All the coding is done from an investigative perspective, asking the participants to connect an LED as their first task. Theycontinue to experiment with the available components to get used to how the code interacts with the devices.

Once they have developed some confidence in coding, participants are

FUNDAMENTAL PRINCIPLES OF **AGILE DEVELOPMENT**

- Only work on the most important things at any given point in time
- Break those things into small bite-sized tasks for individuals to work on autonomously
- Catch up regularly on progress to work out what is important now, and change your plan to adapt if you need to
- Make something simple that works first, then add to it or change it into something better in several steps

given access to promotional or 'call-to-action' videos from UK charities as the basis of their design brief, inspiring them to create something which can help that charity (or its beneficiaries). Once they have selected a charity to support, start-ups then design and prototype their assistive technology through Agile Development Methodology, a process used the world over in technology and software companies.

To teach them the basic tenets of Agile, we run a scaled-back version of agile working practices through the Paper Plane Game. In their teams, young people are asked to create paper planes to varying specifications (and difficulty ratings) in five-minute rounds called sprints, then have their work assessed for its suitability before points are issued for that sprint.

Teams reassess their working plan for the next sprint in a short one-minute stand-up; the winning team is the one with the most points after five sprint/stand-up cycles. While the game sounds simple, it really shows the value for Agile teams in having regular recurring meetings to reassess and adjust their plan (if necessary) to produce a minimum viable product for the client. This format carries on throughout the two days. It makes it feel much more like participants are playing a big role-playing game that's 'refereed' by technically competent, supportive leaders, rather than conventional lesson delivery.

From ideal to real with Agile

The creation and making phase consists of four 90-minute 'sprints', each of which begin with a meeting (or 'stand-up'), just as they would in a real Agile workplace. To maintain the feeling of authenticity and novelty, the prototyping portion of the Hackathon is designed around the concept of a tabletop role-playing game – participants fill and rotate between specific roles every sprint. Each role comes with distinct tasks and responsibilities. These are detailed in the provided guides, and are listed on the lanyards participants wear to indicate which role they're currently playing – these are colour-coded, so facilitators can see at a glance what everyone *should* be doing at any given time.

To maintain the learner-led nature of the Hackathon, all the support required for a total beginner to create their new invention is provided in the form of simple, easy-to-read 'helpsheets'. and a library of videos explaining the function of each available component. Each show a simple wiring diagram and present participants with the simple script required to test the component is working (or have it carry out a basic function). They're encouraged to only use what they need – if they don't need to know about a passive infra-red sensor, why take the time when there are more important things to be getting on with?

As the Hackathon reaches its conclusion, teams are reminded to begin preparing for the final phase of the challenge: the presentation. In order to help the NCS fulfil its goal of encouraging young people to identify and articulate the change they want to make through different activities and phases of the programme (and to measure whether and how they achieved that change), teams reflect upon their experience over the last two days. They create a 'company story' that not only showcases their product but also explains the obstacles overcome, challenges faced, and lessons learned. All teams must present back to the group at large, and be prepared to answer questions about their process and product.

The 2019 delivery of the Social Action Hackathon has just wrapped up in five cities around England; the feedback from participants, facilitators, and stakeholders has been overwhelmingly positive. We've had heartwarming and brilliant inventions, from an internet-connected fall-sensor for the isolated elderly to a motion-sensing nightlight for a children's hospice. It appears that asking young people to engage in coding through social action has shown positive results in overcoming gender barriers to participation in Digital Making, and encouraged personal investment in the creation of something new as well as in the acquisition of important digital skills.

The full library of support materials for the course will be available online once the last bugs are tweaked out of the documents. The suite of reference materials includes a Facilitator Guide (lesson plan and schedule), Developer Guide (student journal and workbook), and a range of technical helpsheets and videos for the most common components and processes (like Bluetooth control, playing sounds with Python and running scripts at boot) used in digital making.

Check future issues of **Hello World** for links to these cutting-edge, world-first resources in the coming months. (HW)

> “ [THEY] DESIGN AND PROTOTYPE THEIR ASSISTIVE TECHNOLOGY THROUGH AGILE DEVELOPMENT METHODOLOGY

MARK 'MR.C' CALLEJA

Resident trickster at The Raspberry Pi Foundation and amateur gentleman-adventurer looking to go pro. Mr.C is a teacher, maker, hacker, Macgyver, Raspberry Pi-certified educator and a quixotic ambassador of geekdom. Since 1983. Twitter: **@M1st3r_C**

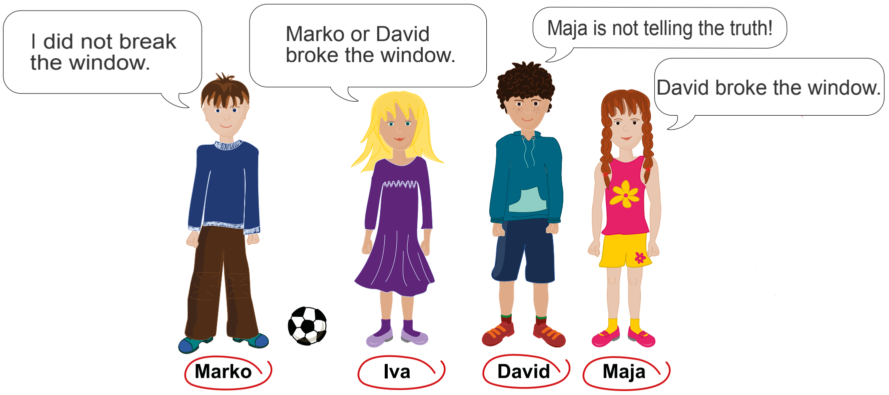
GOING BEYOND THE QUESTION

The question of a broken window – and the computational thinking needed to answer it…

Chris Roffey

When a student takes part in a Bebras challenge, they are presented with a set of attractive computational thinking (CT) questions. However, a lot more than just the question and answer are stored in the Bebras system. In this piece, we look at an example question and see what other information is available. Also, how teachers can access this information to use in their lessons once the challenges are over.

On a beautiful day, Maja, David, Iva, and Marko played football near their teacher Ana's house. Unfortunately, one of them broke her window. Teacher Ana wanted to know who broke the window. Ana knows her students well. She knows that three of them always tell the truth, but she does not know about the fourth one.


I did not break the window.
Marko or David broke the window.
Maja is not telling the truth!
David broke the window.
Marko
Iva
David
Maja

This is what they said:

Question:

Who broke the window?

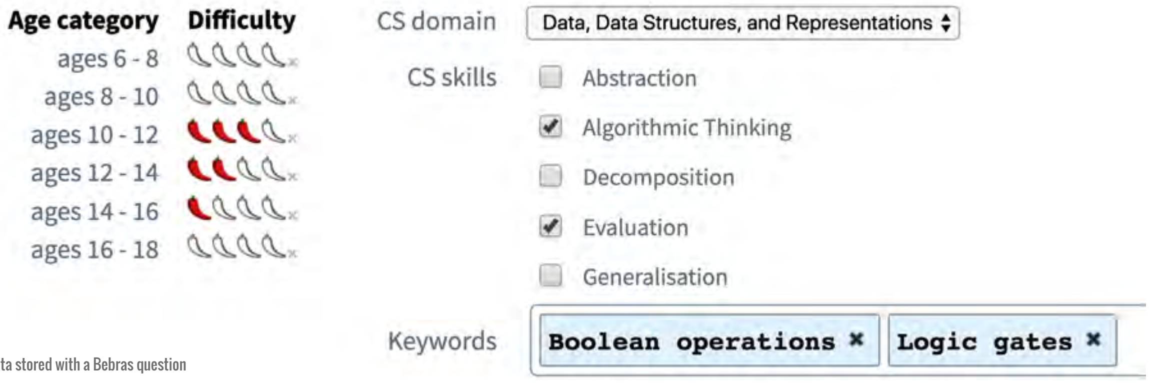
Classification

Each question has associated data that classifies the question in a number of ways. This makes it easier for teachers using the Computational Thinking Quizzes site to quickly find the kind of questions they are after from our archive of hundreds of past questions.

Explanation

Each question also has an explanation associated with it (or in this case, two):

The first thing we notice is that the



Age category	Difficulty
ages 6 - 8	○○○○
ages 8 - 10	○○○○
ages 10 - 12	●●●○
ages 12 - 14	●●○○
ages 14 - 16	●○○○
ages 16 - 18	○○○○

CS domain: Data, Data Structures, and Representations

CS skills:
- ☐ Abstraction
- ☑ Algorithmic Thinking
- ☐ Decomposition
- ☑ Evaluation
- ☐ Generalisation

Keywords: Boolean operations ✖ Logic gates ✖

Classification data stored with a Bebras question

statements from Maja and David cannot both be the truth or both be a lie. Therefore, one of them is telling the truth, and the other is lying. There are two different, equally correct, ways that we could continue. It is important to know that both approawwches exist.

Approach 1:

(a) If Maja tells the truth, then only David lies.

(b) If David tells the truth, then Maja, and either Iva or Marko, are lying, but there can only be one liar in the group.

It follows from these two possibilities **(a)** and **(b)** that David broke the window.

Approach 2:

Alternatively, we could proceed as follows to solve the problem:

(a) If Maja is lying when she says that "David broke the window", then that means the others must be telling the truth. (We know this because Ana knows her students and that three of them always tell the truth.) In that case, Marko is telling the truth when he says that he didn't break the window, and Iva's statement means that David broke the window. But that contradicts David's statement, which means that this cannot be the correct answer.

(b) If David is lying, the others must be telling the truth. In that case, Marko didn't break the window, Iva's statement means that David broke the window, and Maja says the same thing, so this could be a correct answer.

It follows from these two possibilities **(a)** and **(b)** that David broke the window.

Background

Questions also have background information on how it's related to computer science and computational thinking:

The theoretical basis of logical algebra, which underpins all computer programming, was set up in 1854 by George Boole (1815-1864). The basic element of logical algebra is a logical statement. For each statement, it can be unequivocally asserted if it is true or false. In this question, each child was making a valid logical statement.

True statement: true, T or 1
False statement: false, F or 0

Access

Within a week of the end of the official November UK Bebras, we confidentially publish an answer book to registered teachers. This resource supports teachers when their students go through the challenge when it is put into review mode. The answer books are made public later in the year when other countries have finished their Bebras challenges.

The full set of information is also available to teachers at the Computational Thinking Quizzes site, when building their own auto-marking quizzes to use in their schools. 'Quick pick' challenges are also available.

> THAT MEANS THAT TEACHERS CAN PRODUCE CT QUIZZES FOR THEIR CLASSES THROUGH THE YEAR THAT ARE AUTO-MARKED

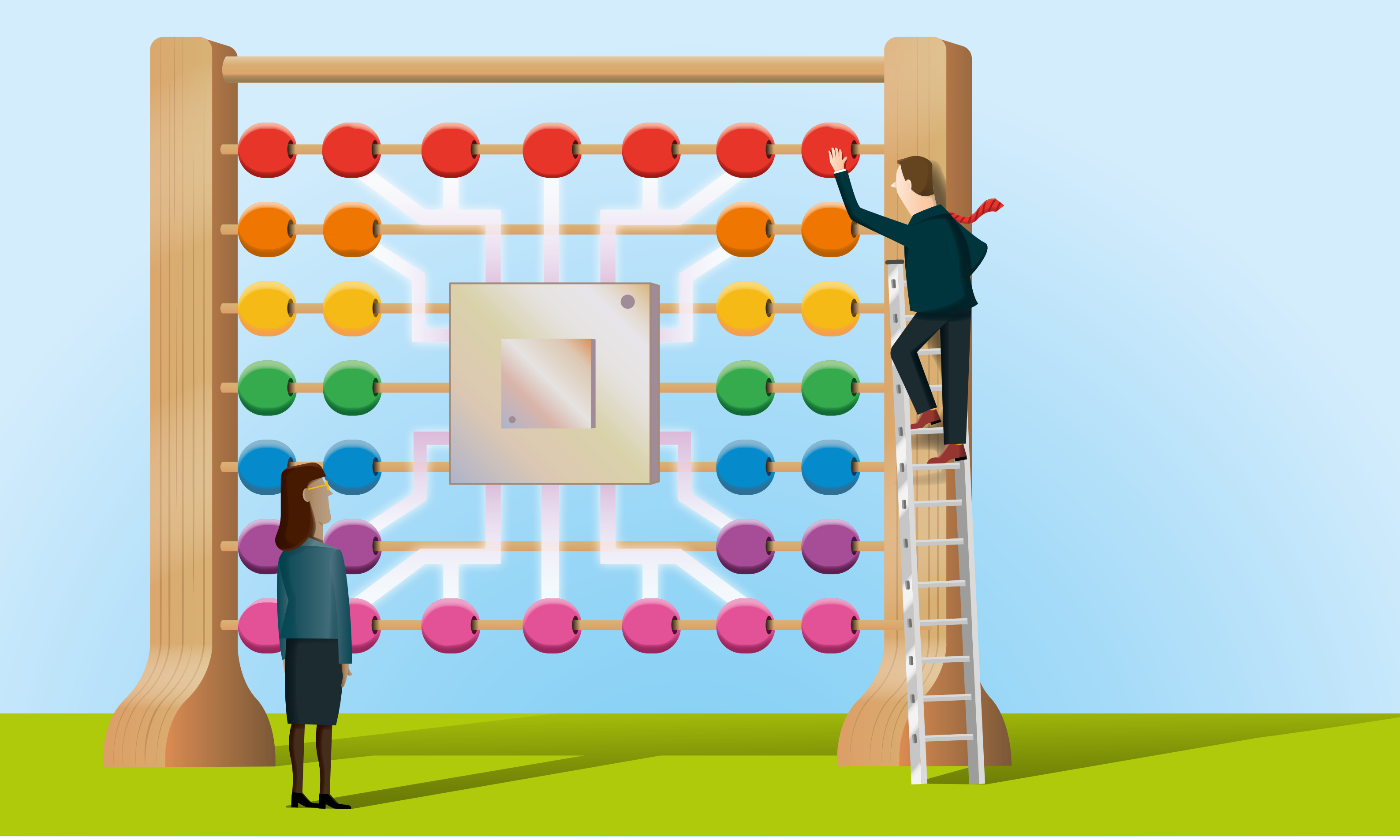
Statements can also be combined in logical terms. Basic logical operations can be created by combining one or two statements. Truth tables can be used to summarise relations between the combined statements. Taking all the statements made by the children together, the students have presented a logical operation that should lead us to who actually broke the window but, in this case, although we know that three statements are True, we do not know about the fourth and, to further complicate matters, the question does not tell us which are the three definitely True statements.

This means that teachers can produce CT quizzes for their classes throughout the year, which are auto-marked and can be put in review mode.

In review mode, teachers can choose whether to show students the explanations and backgrounds when they log back into the quiz, or whether to keep that information to themselves to help support class discussion. (HW)

FURTHER READING

More info: bebras.uk (or email info@bebras.uk)
Answer booklets: bebras.uk/answer-booklets.html
Computational Thinking Quizzes site: quizzes.bebras.uk



COMPUTATIONAL THINKING AND MATHEMATICAL REASONING

Introducing our maths-themed series of features, **Miles Berry** considers the link between computer science and mathematics...

For me personally, mathematics and computer science have always been closely linked. I was first taught BASIC during secondary school maths lessons. My 'O' Level programming project involved calculating lines of best fit and correlation coefficients. Studying mathematics at university, I learnt ML and discrete mathematics through sitting in on CompSci lectures and classes, undertook the optional programming projects, and got summer jobs developing computer models and overhauling Fortran code.

Starting out as a secondary maths teacher, I was keen to integrate technology into my mathematics lessons: I got some of my lessons timetabled for the computer lab, and thus had a good excuse for teaching a little programming for problem-solving in maths. I first learned about Linux and Python in a hands-on maths professional development workshop, ran by the NRICH maths enrichment programme in Cambridge.

Whilst I know there are lots of subjects to which we can connect computing, I remain convinced that there are some particularly powerful synergies that we can exploit when we link CS with maths. There are ample ways to draw on coding in maths lessons – especially so if you've a language that supports functions (such as Snap!, Python, Pyret, as used by Bootstrap World, or even Haskell), and a certain amount of mathematical content that we need to cover in CS teaching (for example, binary and other number bases, Boolean logic, and integer arithmetic), What I'd like to explore here are the strong parallels between computational thinking and mathematical reasoning.

Computations

Before Turing's time, the word 'computer' was actually a job title – folks spent their days performing calculations by hand, following the (sometimes complex) instructions their bosses gave them. I do worry that a lot of school maths seems to be still caught up in this mindset: much of the time seems to be spent getting children to perform routine computations, be they in arithmetic, algebra, or calculus, rather than thinking creatively about solving interesting problems.

At the very foundation of computer science as an academic discipline, Turing sought to establish what it was to do computation: his insight was that this was essentially manipulating symbols on paper according to a set of rules, and that, whilst such computations could be performed by people, they could also be done by machines following the same rules.

> THE CODE OUR STUDENTS WRITE DRAWS ON SUB-SYSTEMS AND FITS IN TO SUPER-SYSTEMS

Over 80 years on, arithmetic, algebra, calculus, geometry, statistics and the rest of maths gets done by computers apart, of course, from in school! For school mathematics to better reflect real world mathematics, shouldn't children spend less of their time doing the work that the machines can do, and more of their time thinking about what rules the machines should follow? In short, shouldn't more of a maths lesson be spent doing computational thinking, and rather less time doing computation?

George Pólya's 1945 book, *How to Solve It* ('A system of thinking which can help you solve any problem'), identified four distinct individual stages to mathematical problem solving:

- Understanding the problem
- Devising a plan
- Carrying out the plan
- Looking back

There are quite clear parallels here with our 'computational thinking': abstraction, decomposition, and generalisation all help with understanding the problem; devising a plan draws directly on algorithmic reasoning; and looking back mirrors evaluation. Carrying out the plan was calculation for Pólya; for us, it's the coding. This approach is a very general one, as Pólya demonstrates in his book, and has stood the test of time well. I'd feel just as confident recommending this as a framework for linking computational thinking and coding in the computer science classroom today as I did for mathematical problem-solving in the maths classroom.

Amongst the many brilliant insights that Seymour Papert shared in *Mindstorms* was that pupils' mathematical reasoning was helped through their learning to code: for Papert, code connected pupils with deep ideas in mathematics at an almost visceral level, although, to be fair, he offered little by way of hard data to support this view. This connection wasn't so much about applying the formulae for the exterior angles of a polygon to draw shapes in Logo, as it was about children discovering the rule through putting oneself in the place of the turtle.

Similarly, I suspect many pupils now form a mental model of four quadrant coordinates through creative play in Scratch way before this gets taught to them in their maths lessons. I could imagine older pupils developing a feel for complex ideas in spatial geometry or statistics through playful, creative engagement with 3D modelling and big(ish) data.

Reasoning

These connections between mathematical reasoning and computational thinking are implicit in the English national curriculum. For example, in computing, 11-14 year olds are taught to:

"design, use, and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems"

Whereas, over in the maths curriculum, we read that pupils,

"can solve problems by applying their mathematics to a variety of routine and non-routine problems with increasing sophistication, including breaking down problems into a series of simpler steps and persevering in seeking solutions.

Abstraction lies at the heart of both computational thinking and mathematical reasoning – in both we capture or model something of the real world – the rules and relationships, the states and behaviours, that lie at the heart of the problems we're solving or the systems we're studying. However, there's nonetheless some difference between how the two domains treat abstraction: in maths, abstraction typically involves getting rid of irrelevant details. Yet in computing, abstraction works in two differing directions: the code our students write draws on sub-systems and fits in to super-systems. Jeanette Wing argued that computational abstraction was more general than its mathematical equivalent, and more concrete, as they are implemented to run on actual, physical hardware.

I'm hopeful that, as our pupils become more fluent with code, they start to look at the problems we set in maths lessons and homework from a coder's perspective. Maybe we might even get to the point where their teachers would accept a program (or an algorithms) to find the solution, rather than an insistence that all working be shown. (HW)

MILES BERRY

Miles (@mberry) is Principal Lecturer in Computing Education at the University of Roehampton. He's a member of the Raspberry Pi Foundation, and serves on the boards of CAS and CSTA.

WHOSE SPLINE IS IT ANYWAY?

How mathematical terminology can obscure otherwise straightforward concepts in **Computer Science**

I didn't take A-level Mathematics, but I really should have done. I'm sorry Mrs Britton, you were right. Starting my Computer Science degree course would have felt less like being asked to walk on stage for the first time and sing an opera to a live audience if I had been better prepared with some mathematical hand-holding beforehand.

It is highly likely that you will remember the first time you tried learning something in the field of computer science and got completely lost. One of my first-year university lecturers insisted on writing elaborate and lengthy Boolean equations on the blackboard, before proclaiming that the proof I had been eagerly waiting for was 'trivial'. I still remember the panic that I should be able to prove it, and the feeling of powerlessness and inadequacy that I didn't know what it was that I didn't know – I didn't even know where to start looking.

When I became a teacher, I vowed to never knowingly make any of my students feel this way about a computer science topic in the classroom. I have been forced to revisit some topics I initially disliked as they are on the A-level specification and – gulp – now I'm the teacher. Every time I have gone to my trusty textbook only to come away with the revelation of, "oh it's just *that*". The thing I was supposed to be learning was indeed relatively easy to grasp, provided you had a friendly and supportive explanation to help you through the mental steps of the initial learning experience.

Terminology and symbols

Mathematics comes with some pretty heavy-duty terminology, which makes it much more difficult for people new to a concept to access or learn about it. Bear with me here, because I'm going to show you how this works. Imagine I've set you this assignment:

"Write a program using a Monte Carlo method for estimating the value of π."

In one sentence I've done two things that (depending on your mathematical background) might potentially cause you considerable anxiety. Do you know what a Monte Carlo method is? Should you? Are you already behind if you don't? I've also included the Greek for pi (π), which you may well be familiar with. But, what if it was a different Greek letter like ∑ or λ? Would you even know how to say that letter out loud? Would you want to ask a question in a lesson about something you weren't even sure how to pronounce?

Here's an explanation of how you would actually go about using this method:

- Imagine a quarter circle with radius 1 inside a 1×1 square
- Generate lots of random points that fall inside the square, like darts thrown blindfold at a dartboard
- You can estimate pi by counting how many points fall inside the circle, compared to the total number of points

A Monte Carlo method is a method that uses repeated random sampling. Here's a

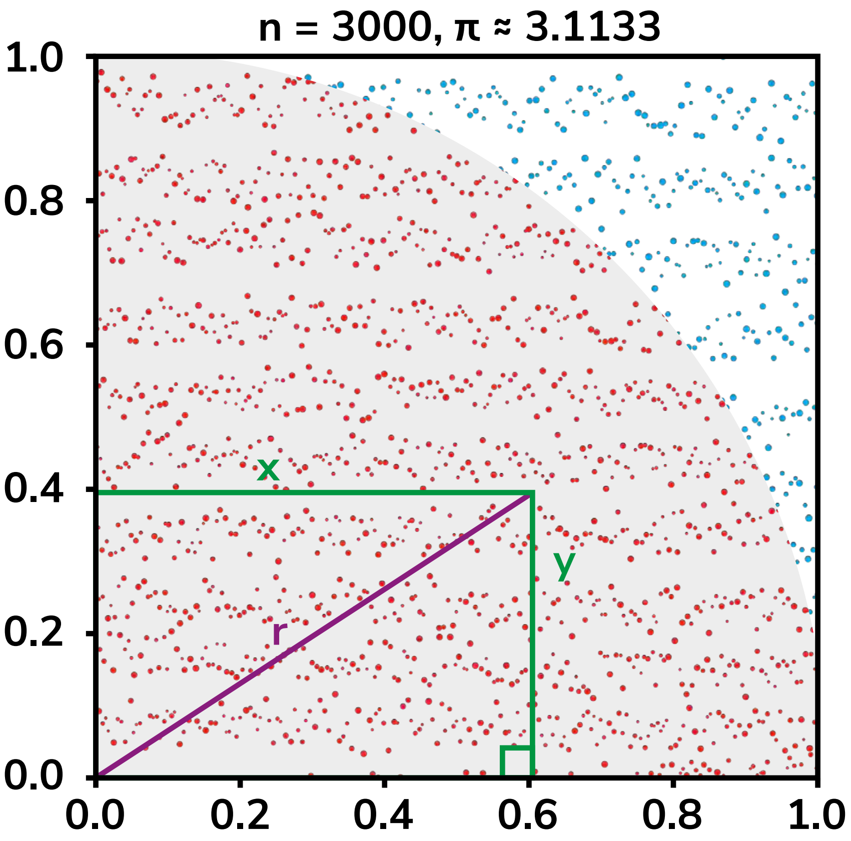


Figure 1: Using random sampling to estimate the value of pi - Adapted from original image by nicoguaro CC BY 3.0, via Wikimedia Commons

picture of what generating those points and drawing them on a graph might look like (see **Figure 1**).

Do you remember Pythagoras' theorem, which is used for calculating the length of the sides of a right angled triangle?

$$a^2 + b^2 = c^2$$

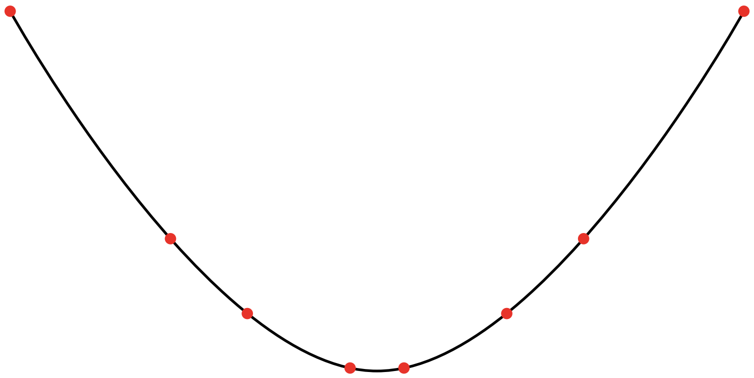
- The indicated point in the diagram has the coordinates x = 0.6 and y = 0.4
- We need to find the distance of this example point from 0,0 to check whether or not it is inside the circle. We'll call this distance r for radius (see the purple line)
- Plug the values of x and y into Pythagoras' theorem to calculate the radius squared:
- $r^2 = 0.6^2 + 0.4^2 = 0.36 + 0.16 = 0.52$
- Finally, take the square root to get the radius. If the result is less than 1, the point is inside the circle. If it is 1 or greater, the point is outside the circle.

Here's the program for generating 100 points and calculating how many are inside the circle:

MATHS IN COMPUTER SCIENCE

Lots of computer science topics studied at school have a natural crossover with mathematics. Here are a few examples:

- Variables - just like in mathematics where students have used x and y (but be careful not to introduce misconceptions)
- Functions - for example in mathematics, f(x) = 2x is a function, it works in exactly the same way as functions we write in a program
- Video game graphics - trigonometry and vectors are used to calculate the movements of a character model. The 'spline' referred to in the title is a curve connecting points on a graph.
- Encryption - your data is protected by a really hard maths problem (finding prime factors of a very large number)
- Machine learning - many modern methods are based on differentiation (and more vectors!)



Splines are useful in video game development

> MONTE CARLO IS A METHOD THAT USES REPEATED RANDOM SAMPLING

LAURA SACH

Laura Sach (@codeboom) is a former Head of Department, and has many years experience teaching Computer Science in the classroom. She now leads the Content Team for Isaac Computer Science.

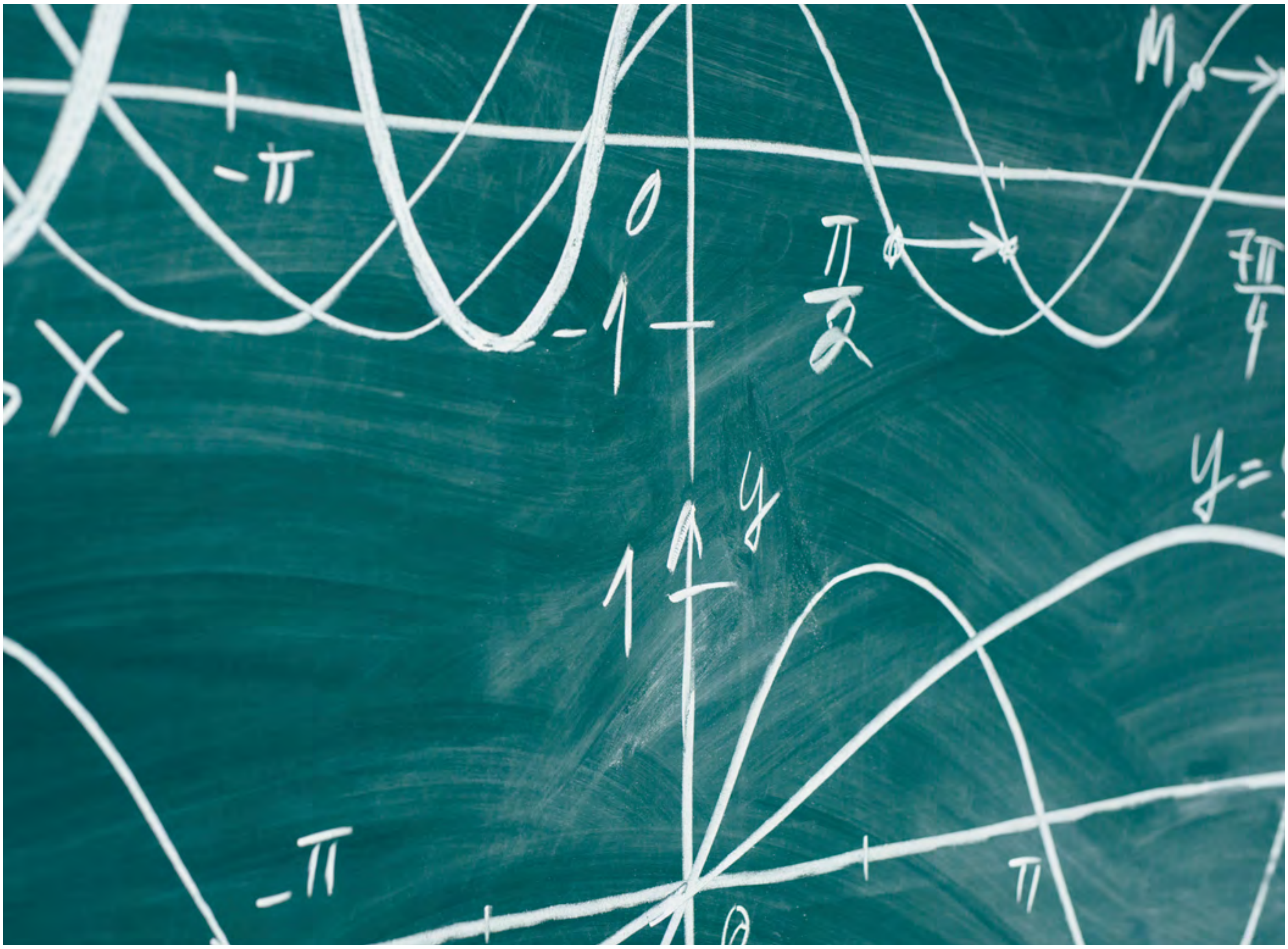
```
inside = 0
n = 100

FOR i FROM 0 TO n
 x = (RANDOM number between
 0 and 1)
  y = (RANDOM number between
  0 and 1)
   result = SQUARE_ROOT(x*x
   + y*y)
  IF result <= 1
    inside = inside + 1
   ENDIF
 ENDFOR
```

What if I had written this article backwards? If I showed you the program first and then introduced how the code uses the Pythagorean theorem, then told you what this was a Monte Carlo method to estimate pi, would you have felt more relaxed about even thinking about this topic? Clearly this depends on your level of mathematical background knowledge, your confidence in programming, your personality, and a whole host of other variables. The point is that this kind of stress can come into play in your classroom accidentally – and if you have a high level of mastery, possibly without you even realising it.

Making maths accessible

I should be very clear here that I am not saying that every mathematical concept is secretly very easy to learn and has just been explained to you badly, because that's not the case. Hard things are hard. However, I think there are some ways that you can make topics which include some mathematical background knowledge much more palatable to learners who are less confident. Here are some things I have tried which work for me:

- Avoid using words and phrases like 'simply', 'just' and 'trivial'. Your trivial might be someone else's nemesis.
- Always call out any background knowledge assumptions in a friendly way. In the topic above I reminded you of Pythagoras' theorem, but I tried hard to make sure you wouldn't feel belittled if you couldn't remember it or its application.
- Don't shortcut bits you think are easy or obvious. If you look at the full resource for estimating the value of pi (**rpf.io/octapi-pi**), taking the square root of r is skipped entirely in the code. This is possible because if a number is less than 1, the square root of that number is also guaranteed to be less than 1 and so it saves a computational operation. However, if you don't know or don't remember this fact, you might be thrown off. I also have not explained in the article why this method works to estimate pi, but the full explanation is in the resource. (HW)



HANDS ON WITH TEACHING SCRATCHMATHS

Examining ScratchMaths as a vehicle for developing computational thinking - and making lessons effective

I'm going to let you in on my secret… I don't have a Computing Science degree. Shocking, isn't it? Thus, when I was asked to become Computing lead, I figured it was because of my interest in linking technology to the main learning taking place in my classroom. However, I learnt very quickly that the programming element, as many of you will be aware, tends to be the 'tricky bit'.

I say 'tricky bit' because it was the bit I had no experience of. Furthermore, it was difficult to navigate staff training and drive up teacher computer science knowledge and understanding whilst the rest of the curriculum and general school life was happening.

Not being a specialist also meant I was pretty much in the dark on how an effective computing lesson should turn out, how it should feel to teach, and how it should be received by the children. More on this later.

However, when I joined a school in Camden, they were already one year into the ScratchMaths project, in partnership with UCL Knowledge Lab and the Institute Of Education (IOE), so I had to get involved pretty sharpish and learn the ropes the best I could. I was shown the resources and talked through with my partner teacher what was happening. I even watched my partner teacher teach the lesson I would take the day after. But there was no avoiding it – I'd come to the project late; I'd missed the initial year of training and was thrown in at the deep end. I was assured by Piers Saunders of the ScratchMaths project that I would be fully supported.

MICHELLE AINLEY

Michelle Ainley is a Computing teacher at Naima Jewish Preparatory School in West London. She is also involved in CAS Westminster.
Michelle tweets at **@computingmissa**.

I was shown all the resources online at **UCL.ac.uk/scratchmaths** and – me being slightly naive – assumed the materials would work like other schemes. I'd read it, copy it into my planning format and review at the end of each lesson how much of the plan was done and what wasn't. Little did I know how wrong my preconceived approach would be!

A few weeks in, Piers offered to come and observe my teaching in a supportive role, the aim being to help provide some insights into my teaching and also how I could differentiate more effectively for my broad Year 6 class. I'd picked up activity 5.1.1 on the presentation, scanned the teacher materials, and off I went.

After the observation, Piers asked how I felt I it went. There were things I clearly needed to work on. My language was mushy, which at the time I didn't realise, but looking back now it was painstakingly obvious. I also tried to complete the lesson at lightning pace (one ScratchMaths activity can last longer than one lesson) and I didn't actually know if the kids fully understood the polygon drawing or not. But I'd got to the end of the plan on the paper so all must be okay, right? Wrong. I felt deflated.

What happened next...

For the next few months, Piers was a great help and came to work with me through observing, intervening, debriefing, and pre-teaching. I began to see the true value of ScratchMaths. I'd read the teacher materials until I could finish each activity, ensuring I understood it on a skills-and-child-level, also ensuring I could identity and navigate a few hiccups or misconceptions. I'd also bought Carol Vorderman's *Computer Coding for Kids: A Unique Step-by-Step Visual Guide* as a reference point, as I didn't quite know what a variable was at that point. This also helped improve my vocabulary, so I could improve the children's computational articulation too, something which I feel is very important for computing. Yes, I did have the support of Piers, but I do feel that through following the teaching materials, being reflective and taking your time, ScratchMaths can be extremely powerful for teaching programming and achievable by all.

Undeniably, Scratch is a great resource but I do feel it is a bit of a buzzword thrown around by a number of teachers who feel it has something to do with coding.

Module 1 of ScratchMaths addresses the very basics and allows children to learn about the environment in Scratch, something which can often be overlooked in lesson planning. I want the students I teach to progress to secondary school and be able to articulate in a manner which other students from other schools will understand, as I believe this is a key concept not only in programming, but in life. ScratchMaths lessons significantly contribute to the development of programming vocabulary, which I found useful teaching in a diverse setting.

For now, the teacher materials are still

designed for Scratch 2.0 but most files can be run on Scratch 3.0 and I have heard mumblings that the guidance may be updated for 3.0 soon. Also, Scratch for Education makes managing student accounts so much easier.

I am fortunate that I now know the scheme well enough to use it effectively and mould it how I wish. I continue to dabble in PRIMM (the work of Sue Sentance) and I am continuing to find PRIMM opportunities with the modules in ScratchMaths. ScratchMaths has some great activities which can be used for predicting and I have worked to fully embed this into my school's curriculum to show my students that computational thinking happens beyond a computer.

ScratchMaths is also great for differentiation. The support materials allow me to extend my more able students with challenges, and provide additional guidance to those who are progressing at a slower pace. ScratchMaths has also supported my planning of STEM workshops, where my students can transfer skills from ScratchMaths to other contexts and challenges.

> MY SUBJECT KNOWLEDGE HAS IMPROVED TENFOLD SINCE I STARTED USING SCRATCHMATHS

In practice

Activity 5.1.2 of Module 5 is one of my absolute favourites for Year 5 or 6, as it generates great paired discussions. On introduction, the students often think the scripts of code deliver the same outcome due to the blocks that have been used, so they need to use computational thinking skills to determine the differences. They also become engrossed in finding out what the outcome actually is. But it's also important to model how to properly predict what the script will achieve. I often allow the children a few minutes to tackle it themselves and judge if anyone has gone off track or is rather lost, then bring them back to model how to deconstruct the scripts. This can require a bit of time and, sometimes, if misconceptions emerge, this activity can last a full lesson, that I end up following up in consecutive weeks. But it works well with a PRIMM approach and my students enjoy running the codes before modifying them to create different polygons.

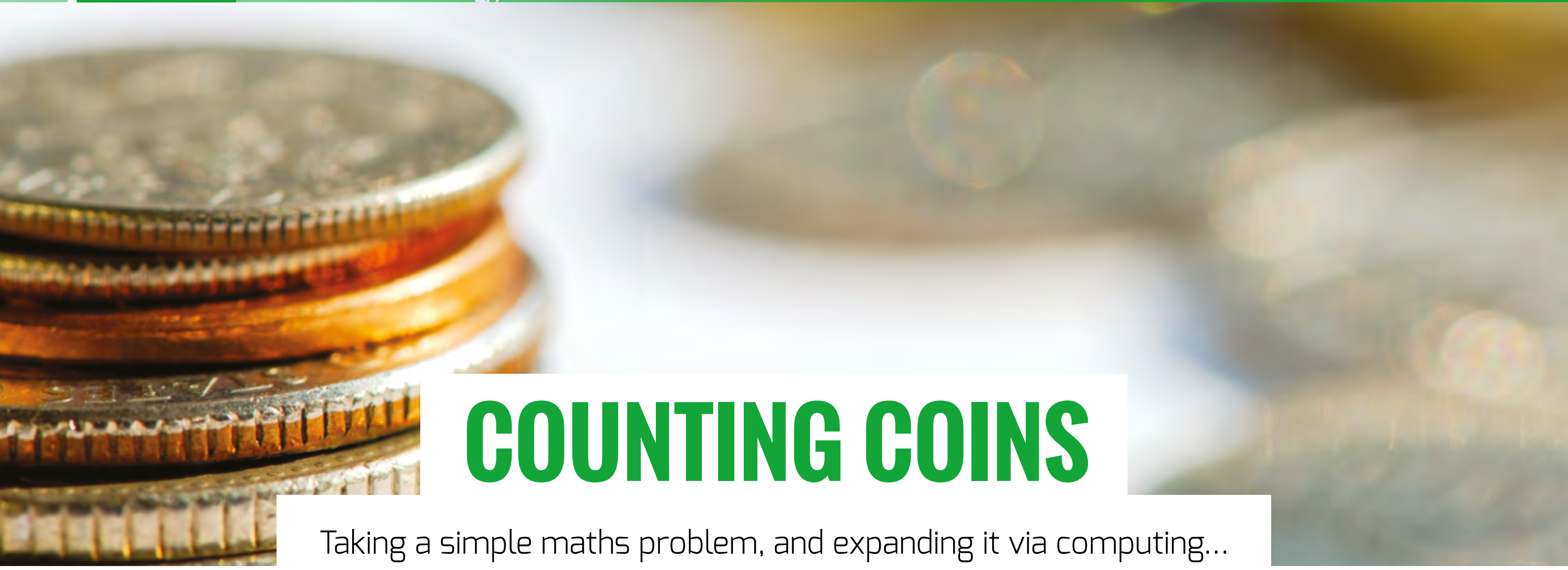
I have also been able to link activity 5.1.2 to coding in Python, and I have incorporated drawing polygons using Turtle.

In terms of developing subject knowledge, I have established teaching triads throughout Key Stage 2 (ScratchMaths is designed for Years 5 and 6 but I have started it in Year 4 to counteract the effects of SATs in Year 6). Three teachers work together by informally observing each other, discussing the same lesson, and exploring the content of the guidance materials. I also team-teach to support those who may be more resistant to, or have lower confidence in, teaching computing.

I've never enjoyed a scheme of work, purely because I've always felt it crushes my creativity and autonomy, and puts pressure on me to deliver a lesson. But ScratchMaths is different. As I said at the beginning, I don't have a Computer Science degree, but I've learnt that that is okay. My subject knowledge, including my understanding of computational thinking, has improved tenfold since I started using ScratchMaths. I articulate computer science concepts much more accurately and this is shown in the discussions which take place in my classroom. As a resource (which is also free!), I am yet to find anything which delivers computational thinking activities and predicting activities quite like it, whilst also fully supporting the teacher and developing subject knowledge.

I was concerned that the children may not be fully invested in ScratchMaths, due to its more formal nature, but I quickly learnt that like other lessons, my enthusiasm sets them up for their investment and actually their feedback has been extremely positive.

The final report may not have found a link between improving mathematics in relation to the end of Key Stage 2 SATS, but in my own experience, ScratchMaths has improved teacher subject knowledge, computational thinking, problem-solving, and my student's understanding of block coding in Scratch. From a computing point of view, that's pretty good! (HW)



COUNTING COINS

Taking a simple maths problem, and expanding it via computing…

Somewhere close to the start of the English maths curriculum, pupils learn to work with coins, initially as physical manipulatives, then as icons and eventually as numbers. By the age of seven or eight, pupils are taught to "find different combinations of coins that equal the same amounts of money". This sounds quite straightforward.

Finding the smallest number of coins to make a particular amount of money is something we do without consciously thinking about it, but if you had to teach a child to do this, what method would you suggest? It's something which machines can do too: vending machines give change; good vending machines give change using the smallest number of coins they can. How would you program a vending machine to do that?

Algorithm

It's a problem that lends itself to a greedy algorithm:

- Start with the largest value coin
- Keep giving that coin until the amount left is less than the value of the coin
- Move on to the next largest value coin and do that again
- Keep moving on to the next largest value of coin until there are no more coin values left

Following this algorithm by hand seems to give the answers we expect. E.g. 38p change: 20p, 10p, 5p, 2p, 1p.

A naive approach to coding this in Scratch might look a little like **Figure 1**.

As pupils become more fluent with programming in general and Scratch in particular, they might make use of more procedures for dealing with each coin value in turn or perhaps lists for the coin values and the coins needed **(Figure 2**).

Whilst the maths here is on Year 2 of the primary curriculum, I don't think we would expect Year 2 to be able to write Scratch code to work out the solution. By the end of primary, however, I'd hope that most pupils would be able to create a version of this in Scratch themselves.

At some point in Key Stage 3, I'd hope that pupils could make the transfer over to coding the same algorithm in Python, perhaps along these lines:

```
def fewestCoins(amount,
values=[200, 100, 50,
20, 10, 5, 2, 1]):
coins = []
 for value in values:
   while amount >= value:
     coins = coins + [value]
     amount = amount - value
return coins
```

This works well enough for English coins. We can generalise this, though: it's easy enough to modify it to work with US coinage too, with English pre-decimal currency, and with the equally Byzantine Galleons, Sickles, and Knuts if you find yourself teaching programming at Hogwarts.

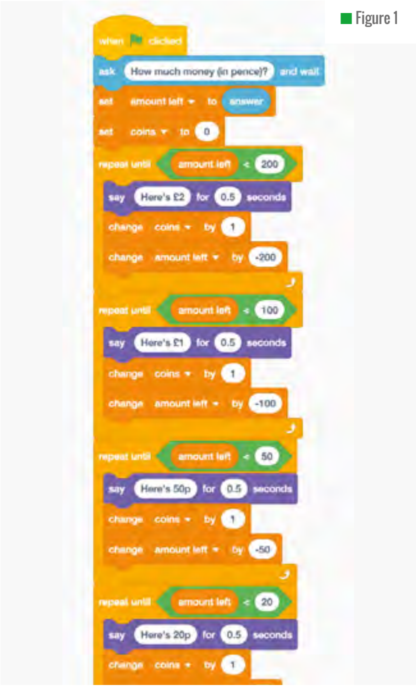


Figure 1

```
when 🏴 clicked
ask (How much money (in pence)?) and wait
set [amount left ▾] to (answer)
set [coins ▾] to (0)
repeat until <(amount left) < (200)>
  say (Here's £2) for (0.5) seconds
  change [coins ▾] by (1)
  change [amount left ▾] by (-200)
end
repeat until <(amount left) < (100)>
  say (Here's £1) for (0.5) seconds
  change [coins ▾] by (1)
  change [amount left ▾] by (-100)
end
repeat until <(amount left) < (50)>
  say (Here's 50p) for (0.5) seconds
  change [coins ▾] by (1)
  change [amount left ▾] by (-50)
end
repeat until <(amount left) < (20)>
  say (Here's 20p) for (0.5) seconds
  change [coins ▾] by (1)
```

Bit coins?

But look, it works well enough for this set of coins too: 128, 64, 32, 16, 8, 4, 2, 1. That is, the binary place values for eight bits. For example:

38p = 32p + 4p + 2p.

99p = 64p + 32p+ 2p + 1p.

And so on. Given the ease with which pupils can find the smallest number of coins for amounts, I think a set of, let's call them, bit-coins, perhaps first as manipulatives, then as pictures, finally as numbers, could be the easiest way to get pupils converting numbers to binary.

The earlier Python code works well enough for working out the bit values, and could be used for other number bases too – use 64, 8 and 1 to get octal. Use 16 and 1 to get hexadecimal. I think a coin-based model such as this goes a long way to internalise an understanding of the maths here.

If we'd like to actually get the binary out, then a little remixing is needed.

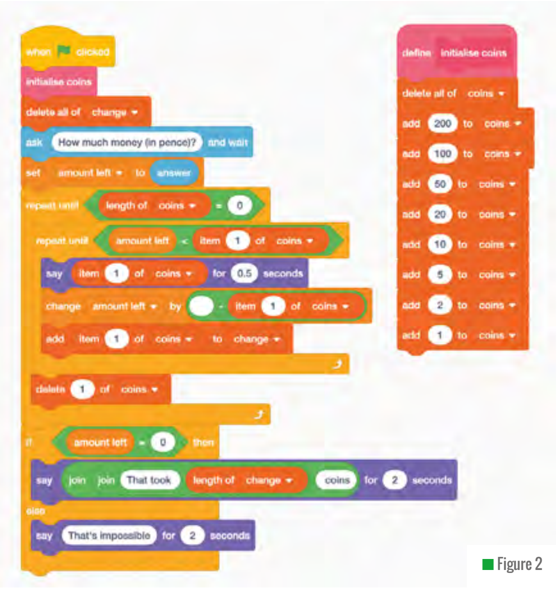
```
def decimalToBinary(decimal):
   bitValues = [128, 64, 32, 16,8,
   4, 2, 1]
   bits = “”
   for value in bitValues:
      if decimal >= value:
         bits = bits + “1”
         decimal = decimal - value
      else:
         bits = bits + “0”
   return bits
```

However...

Looks great, doesn't it? We've got a general algorithm here to solve all sorts of coin decomposition problems, and have generalised it to convert numbers from to binary and even to other number bases.

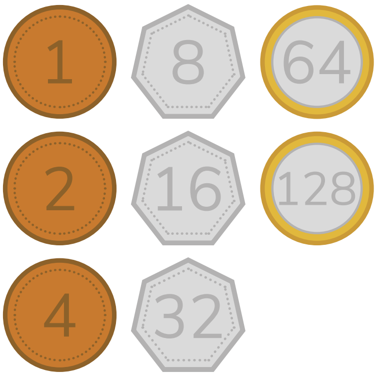
But it doesn't always work.

What about coins worth 1p, 10p, 20p and 25p? How would you make 40p? If you follow the greedy algorithm, you'd use a 25p, 10p and five 1p coins. But if you stop and think, just two 20p coins would be better! How come it breaks down here?



```
when [flag] clicked
initialise coins
delete all of [change ▾]
ask [How much money (in pence)?] and wait
set [amount left ▾] to (answer)
repeat until <(length of [coins ▾]) = (0)>
   repeat until <(amount left) < (item (1) of [coins ▾])>
      say (item (1) of [coins ▾]) for (0.5) seconds
      change [amount left ▾] by (() - (item (1) of [coins ▾]))
      add (item (1) of [coins ▾]) to [change ▾]
   end
   delete (1) of [coins ▾]
end
if <(amount left) = (0)> then
   say (join (join [That took] (length of [change ▾])) [coins]) for (2) seconds
else
   say [That's impossible] for (2) seconds
end

define initialise coins
delete all of [coins ▾]
add (200) to [coins ▾]
add (100) to [coins ▾]
add (50) to [coins ▾]
add (20) to [coins ▾]
add (10) to [coins ▾]
add (5) to [coins ▾]
add (2) to [coins ▾]
add (1) to [coins ▾]
```

Figure 2



Or, what if we only have 7p and 9p coins. What do you use to make 47p? My code happily returns five 9p coins, but that's wrong! You can't make 47p using just 7p and 9p 'coins'. Why not? You can, however, make any amount bigger than 47p, although my greedy algorithm gets most of these wrong too.

Coming up with an algorithm that works for these situations too is more tricky. One idea relies on the recursive nature of the problem. Sticking with English coinage, to make 38p, you could start with a 20p and make 18p, or start with 10p and make 28p, or start with 5p and make 33p, or start with 2p and make 36p, or start with 1p and make 37p. So the smallest number of ways to make 38p has to be one more than the smallest number of ways to make 18p, 28p, 33p, 36p or 37p, whichever is least.

Our recursive rule here looks like:

```
minimumcoins(x) = 1 + minimum (
minimumcoins(x-200),
minimumcoins(x-100),
minumumcoins(x-50),
minimumcoins(x-20),
minimumcoins(x-10),
minimumcoins(x-5),
minimumcoins(x-2),
minimumcoins(x-1))
```

With the base cases that minimumcoins for 200, 100, 50, 20, 10, 5, 2 and 1 are all 1, as the smallest number of coins needed to make 50p is pretty trivial!

Generalising this for any set of coin values gives us this recursive Python code:

```
def fewestCoinsRec
(amount, values=[200,
100, 50, 20, 10, 5, 2,
1]):
   minCoins = amount
   if minCoins in values:
      return 1
   else:
      for thisCoin in [value for
      value in values if value <=
      amount]:
   numCoins = 1 +
   fewestCoinsRec(amount -
   thisCoin, values)
      if numCoins < minCoins:
         minCoins = numCoins
return minCoins
```

But this is very slow – for instance, my 38p with English coins example takes over 37 million calls to the recursive function, and takes nearly 30 seconds to run on my MacBook. I suspect most Year 2 children would be quicker!

We can do things to speed this up – using a dictionary to keep track of the minimum coins found so far for each amount we look up helps. If we keep track like this, it's faster to build up the dictionary from the bottom up – with English coins, you need 1 coin to make 1p, 1 coin to make 2p, 2 coins to make 3p, 2 coins to make 4p, 1 coin to make 5p, and so on. To find out how many coins you need to make 6p, you need to take the minimum from making 5p (1), 4p (2) or 1p (1) and add a coin to that: that's two coins, as you'd expect. I'll leave writing the code for that as an exercise...

So, we've a simple Year 2 maths problem, but taking this into computing, we can automate solving it, we can generalise this to a far broader class of problems, we can link this to binary conversion, we can get to grips with some logical bugs, and apply sophisticated techniques like recursion. From this example alone, I reckon it's worth making these connections between the maths curriculum and computing! (HW)

MILES BERRY

Miles (@mberry) is Principal Lecturer in Computing Education at the University of Roehampton. He's a member of the Raspberry Pi Foundation, and serves on the boards of CAS and CSTA.

LAMBDA CALCULUS WITH THE AMAZING ALONZOS

The Amazing Alonzos are a troupe of conjurors, whose acrobatic antics are curiously like the lambda calculus

In this two-part article we'll look at Alonzo Church's lambda (λ) calculus. It can look quite daunting at first, but we've got some acrobats to help us! The λ calculus is, along with Turing machines and recursive function theory, one of the pillars of modern computer science. It's particularly important as the basis of functional languages like Haskell and OCaml, and for the wider use of anonymous and higher order functions, now found in Python and Java.

Instead of starting with λ calculus, in this first part we're going to watch The Amazing Alonzos, a troupe of acrobatic conjurors. As it turns out, the Alonzos are a simple model for computation, somewhere between Church's λ calculus and the combinatory logic developed by Church's mentor Haskell Curry.

Figure 1 shows four key troupe members. Note that the acrobats are very shy, so we only ever see them from the back. The Alonzos have the ability to form human pyramids of arbitrary height, which can change shape and composition depending on who's balanced on who. Pyramids can get bigger as well as smaller, and some pyramids can make copies of their members.

Let's start with Ida, who turns into the pyramid she holds on her left hand. For example, Ida balancing Trudy turns into Trudy. See **Figure 2 (a)**. Next, let's meet Trudy. Trudy can balance with both hands, and turns into the pyramid on her left hand, so Trudy balancing Ida and Falko turns into Ida.

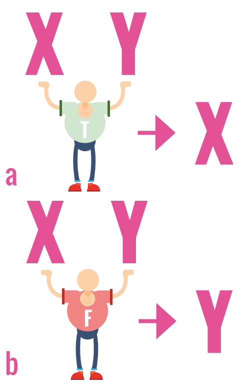


Figure 4: (a) Trudy as true (b) Falko as false

MEET THE AMAZING ALONZOS

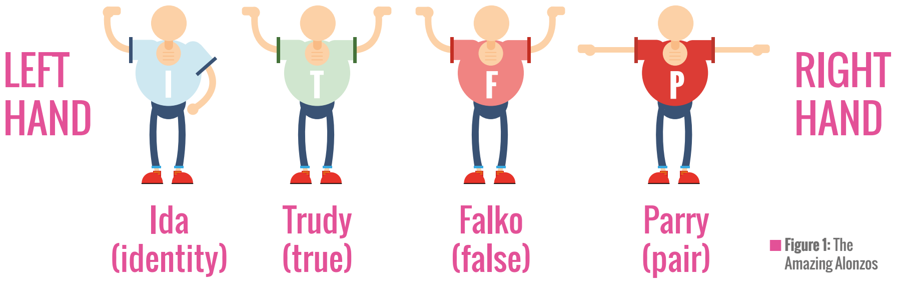


Figure 1: The Amazing Alonzos

See **Figure 2 (b)**. And let's meet Falko. Like Trudy, he can balance with both hands, but, unlike Trudy, he turns into the pyramid on his right hand. So, Falko balancing Parry and Trudy turns into Trudy. See **Figure 2 (c)**.

Finally, let's meet Parry, who can balance pyramids on both hands, and on his head. When he's fully loaded, he changes into the pyramid on his left hand balancing the pyramid on his head and right hand. So, Parry balancing Trudy, Falko, and Ida turns into Trudy balancing Falko and Ida, which turns into Falko. See **Figure 3**.

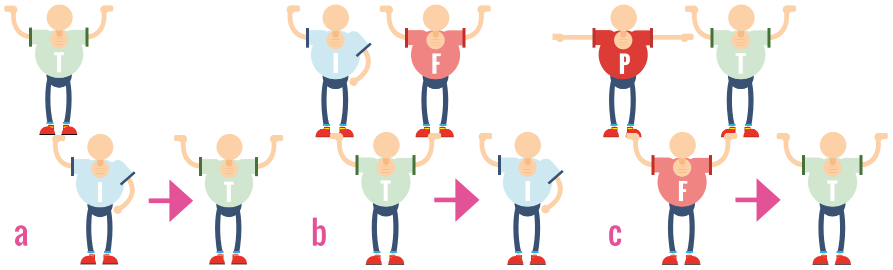


Figure 2: (a) (Ida Trudy) => Trudy (b) ((Trudy Ida) Falko) => Ida (c) ((Falko Parry) Trudy) => Trudy

Logic with The Amazing Alonzos

What's this got to do with Computer Science, let alone λ calculus? Well, suppose that an Alonzo pyramid is a program, and the members of the troupe are the elementary instructions. A changing pyramid is then like an executing program. As we shall see, this is more than just an analogy: an Alonzo pyramid really is a program manipulating data.

The difficulty is that we can't immediately identify the sorts of things that our familiar programs work with, like values, and control and data structures. However, we can represent all of these using different Alonzo pyramids.

The key is Parry, who does things to pairs of pyramids. For sure, he does things with three pyramids but, as we'll see, he does the first pyramids to the other two pyramids.

Let's start by thinking about a conditional expression:

IF ? THEN X ELSE Y

When the condition ? is true, the THEN option X is selected; when it's false, the

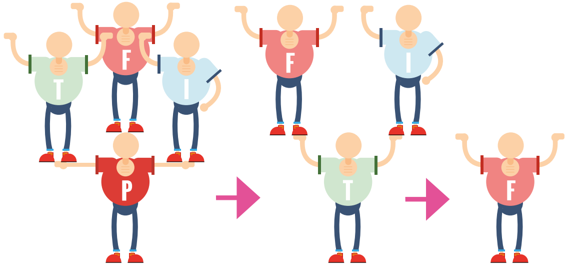


Figure 3: (((Parry Trudy) Falko) Ida) => ((Trudy Falko) Ida) => Falko

ELSE option Y is selected. So true selecting X is just like Trudy selecting her left hand, and false selecting Y is just like Falko selecting his right hand. See **Figure 4**.

This suggests that we can build a conditional expression straight out of Parry. See **Figure 5**. We can also use Parry to represent logical operations like AND and OR. See **Table 1**.

AND is the same as Y when X is true, and false when X is false, so we can make AND out of Parry balancing X, Y, and false. See **Figure 6**. OR is true when X is true, and the same as Y when X is false. So we can make OR out of Parry balancing X, true, and Y. See **Figure 7**.

Indeed, we can make all the binary operators, and hence any boolean expression, out of Parry, Trudy, and Falko. Furthermore, with variants of Parry, we can represent integers and lists, building up to a whole programming language, but we won't consider that here.

In the second part of this article (that'll be appearing in the next issue) we'll firm up the link from the Alonzos to λ calculus, by writing them as anonymous functions in Python. In the meantime, you can find a more thorough treatment of λ calculus itself in my ageing book, *Introduction to Functional Programming Through λ Calculus*, which is available as a PostScript at: **macs.hw.ac.uk/~greg/books**. (HW)

GREG MICHAELSON

Greg is Emeritus Professor of Computer Science at Heriot Watt University. His research interests are in the design, implementation, and analysis of programming languages, and in programming education. He has taught programming since 1977.

X	Y	X and Y	X or Y
False	False	False	False
False	True	False	True
True	False	False	True
True	True	True	True

Table 1: Truth tables for AND and OR

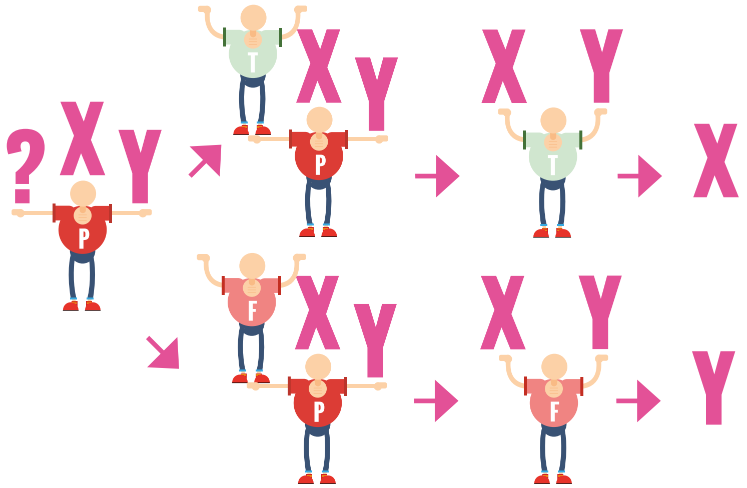


Figure 5: Parry as IF-THEN-ELSE

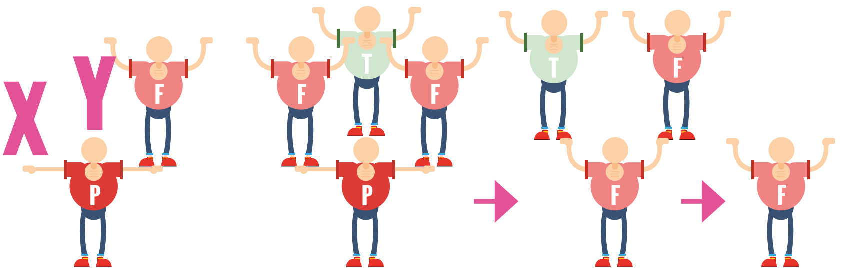


Figure 6: (a) Parry as X AND Y (b) false AND true => false

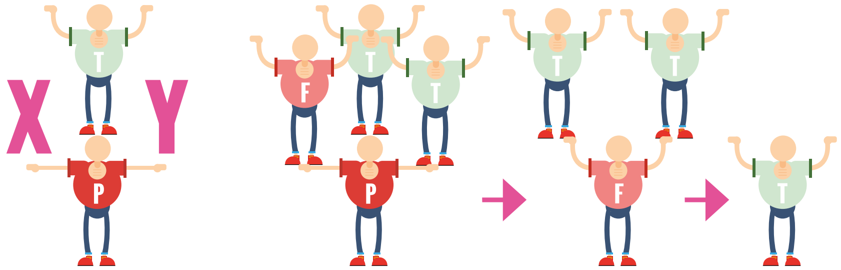


Figure 7: (a) (a) Parry as X OR Y (b) false OR true => true

#INSIGHTS

PRAXIS: BRINGING TOGETHER THEORY AND PRACTICE

STORY BY Oliver Quinlan

In any subject with a practical element, people sometimes feel a tension between theory and practice. When both come together, we can unlock the power of praxis for learning.

Theory and teaching

Having a theoretical framework for thinking about teaching and learning is as important as having an answer to the question: 'Why do you want to be a teacher?' That question invites you to articulate your moral purpose, and a theoretical framework is the place to articulate your intellectual purpose. It shapes the thinking you do, the decisions you make, and the paths your learners take. This framework can change, of course. It will continue to be refined and influenced by experience, reading and thinking.

What about experience?

Your theoretical framework influences your practice, but your experience in the classroom also continues to shape your framework; the two are not separate. In fact, the Brazilian educator and theorist Paulo Freire used a different term to describe this unity between theory and practice: praxis.

Praxis

Praxis could be summed up as 'informed action' – it is the process of taking action in practice whilst acting within a theoretical framework of thought. In this concept, theory and practice are as one; for Freire's revolutionary politics did not make a distinction between the importance of thinking differently and the importance of making a concrete change in the world. In praxis, abstract theorising is only useful so long as it informs concrete action. Likewise, action must be informed by deep thinking and justification. Only in this way did Freire see 'the oppressed' finding their own, new way to intellectual and social freedom, rather than simply repeating the mistakes of their 'oppressors'.

Praxis in practice

The notion of praxis is that you shouldn't 'do' and then 'reflect' on it later, rather make sure every action has an informed basis, whilst you put every valuable thought into action. A teacher involved in bringing theory to practice could consider their actions when planning, then again when reflecting. A teacher immersed in praxis would bring their theoretical thoughts to every decision as they make it. They adapt their actions in a classroom to ensure they continue to encourage the learning their students are undertaking. It situates the learning as a conversation between learner and teacher, rather than a teacher carrying out plans which were crafted in the hypothetical world of being 'good in theory'.

Teaching is a complex business of both practical action and intellectual consideration. So often we define these two facets as distinct, when we would do well to heed Freire's call to 'informed action'. (HW)

FURTHER READING

- ✓ Croll, P. and Hastings, N. (1990). *Effective primary teaching: research-based classroom strategies* (Letts)
- ✓ Freire, P. (1970). *Pedagogy of the Oppressed* (Penguin)

USING NON-PROGRAMMING ACTIVITIES TO TEACH PROGRAMMING CONCEPTS

STORY BY Thom Kunkeler

Programming takes place on a computer, but research has shown the promise of using non-programming activities for teaching important concepts to novice learners. The study, conducted by Grover et al at three urban US schools, was designed with 16 non-programming activities during a 20-day programme. The research team found that the learning gains from students who followed the intervention were significantly higher, in comparison to students who followed the regular computer science curriculum.

> “ STUDENTS HAD TO… RESCUE THE CATS

From the moment learners engage with any programming language or tool, they will encounter variables, expressions, loops, and abstraction – the 'VELA' concepts – in one way or another.

Some learners struggle with these concepts, as they assume that variables can have multiple values at the same time, whereas others might not be able to distinguish between what goes inside a loop and what precedes a loop. Although visual block-based programming environments such as Scratch, Alice, or Snap! have helped with navigating through different concepts, students still find it difficult to make sense of programming languages.

For this study, the research team designed non-programming activities to help students engage with and understand the foundation of programming concepts. Activities such as 'Story Variables' and 'Cats and Ladders' were designed to encourage students to come up with a definition of a 'variable' collaboratively, before applying this definition in practice. In the first activity, students discussed multiple examples where variables were present, such as: “Last week, I bought a pen for $1.50, now it costs $3,” to identify what the variable is, and how it changes over time. In the second activity, the students had to put their definition into practice by determining the length of a ladder required to reach distraught cats in 'Cats and Ladders'.

Students had to come up with a range of possible values that were needed to rescue the cats, and engage in abstraction by synthesising new variables based on existing ones. (HW)

FURTHER READING

Full details on the article and activities can be found here:

https://bit.ly/2malUFL

Examples of activities	Description
Story variables	Introducing ideas of changing quantities in the real world and giving them meaningful names
Cats and Ladders	Naming variables and creating expressions in non-programming context
3 switches	Turning light switches on/off using Booleans, Boolean operators and abstractions
Alarm clock	Modelling real-world alarm clock situations using variables, arithmetic, and logical expressions and abstractions

EXAMPLES OF NON-PROGRAMMING ACTIVITIES FOR THE CLASSROOM

Throughout the 20-day research intervention, students participated in 16 of such activities. The research team gathered their data based on a pre- and post-assessment of students' introductory programming skills, analysis of students' Scratch projects, and interviews with teachers and students alike. Besides higher learning gains for students following the intervention, the findings were not linked to gender or prior academic preparation, and all grades participating in the study showed similar learning gains. For teachers, these findings should be taken as an incentive to experiment with non-programming activities for teaching difficult concepts. Try out some of these non-programming activities yourself!

SUBSCRIBE

SIGN UP TODAY FOR **FREE**

THE MAGAZINE FOR COMPUTING & DIGITAL MAKING EDUCATORS
(Hello World)
Issue 10
Oct 2019
helloworld.cc
WHY TEACH CHILDREN TO CODE?
The role of programming in CS
CYBERSECURITY IN CLASS
Where to get started
PLANNING THE PERFECT OPEN EVENING
Insider's guide inside
SORTING OUT SHAPES
Logo lesson tips
LET IT GROW!
When technology meets the school garden
SEMANTIC WAVES
Ideas for reviewing lesson activities
MATHS & COMPUTER SCIENCE
Fresh ideas and activities for your classroom
PLUS
BRINGING PHYSICAL COMPUTING TO THE CLASSROOM • MAKING NEWS AT SKY • CODE CLUBS FOR TEACHERS
RETRIEVAL PRACTICE & LEARNING • GOING BEYOND THE QUESTION • IMAGINING THROUGH VIRTUAL REALITY

FREE IN PRINT
For UK-based educators!

SUBSCRIBE TODAY

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

USING DATA TO HELP A SCHOOL GARDEN

How a school in New Jersey is directly linking data with life itself...

Every year, our students take federal or state mandated testing, but what significant changes have we made to their education with the results of these tests? We have never collected more data about our students and society in general. The problem is most people and institutions do a poor job interpreting data and using it to make meaningful change. This problem was something I wanted to tackle in FH Grows.

FH Grows is the name of my seventh-grade class, and is a student-run agriculture business at Knollwood Middle School in Fair Haven, New Jersey. In FH Grows, we sell our produce both online and through our student-run farmers markets. Any produce we don't sell is donated to our local soup kitchen. To get the most out of our school gardens, students have built sensors and monitors using Raspberry Pis. These sensors collect data which then allows me to help students learn to better interpret data themselves and turn it into action.

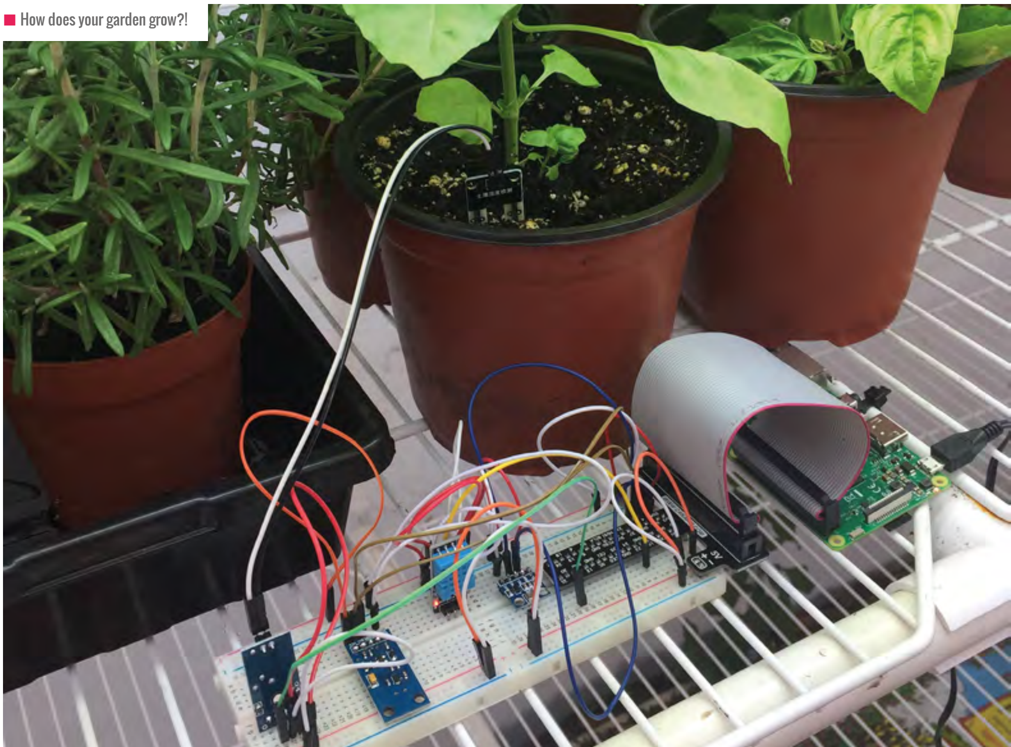
Turning data into action

In the greenhouse, our gardens, and alternative growing stations (hydroponics, aquaponics, aeroponics) we have sensors that log the temperature, humidity, and other important data points that we want to know about our garden. This data is then streamed in real time, online at **FHGrows.com**. When students come into the classroom, one of the first things we look at is the current, live data on the site and find out what is going on in our gardens. Over the course of the semester, students are taught about the ideal growing conditions of our garden. When looking at the data, if we see that the conditions in our gardens aren't ideal, we get to work.

If we see that the greenhouse is too hot, over 85 degrees, students will go and open the greenhouse door. We check the temperature a little bit later, and if it's still too hot, students will go turn on the fan. But how many fans do you turn on? After experimenting, we know that each fan lowers the greenhouse temperature between 7-10 degrees Fahrenheit. Opening the door and turning on both fans can bring a greenhouse than can push close to 100 degrees in late May or early June down to a manageable 80 degrees.

Turning data into action can allow for some creativity as well. Over-watering plants can be a real problem. We found that our plants were turning yellow because we were watering them every day when we didn't need to. How could we solve this problem and become more efficient at watering? Students built a Raspberry Pi that used a moisture sensor to find out when a plant needed to be watered. We used a plant with the moisture sensor in the soil as our control plant. We figured that if we watered the control plant at the same time we watered all our other plants, when the control plant was dry (gave a negative moisture signal) the rest of the plants in the greenhouse would need to be watered as well.

This method of determining when to water our plants worked well. We rarely ever saw our plants turn yellow from overwatering. Here is where the creativity came in. Since we received signal from the Raspberry Pi when the soil was not wet enough, we played around with what we could do with that signal. We displayed it on the dashboard along with our other data, but we also decided to make the signal send as an email from the plant. When I showed students how this worked, they decided to



How does your garden grow?!

write the message from the plant in the first person. Every week or so, we received an email from Carl the Control Plant asking us to come out and water him!

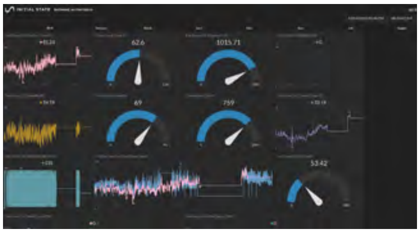
If students don't honour Carl's request for water, use data to know when to cool our greenhouse, or had not done the fan experiments to see how much cooler they make the greenhouse, all our plants, like the basil we sell to the pizza places in town, would die. This is the beauty of combining data literacy with a school garden: failure to interpret data then act based on their interpretation has real consequences: our produce could die. When it takes 60-120 days to grow the average vegetable, the loss of plants is a significant event. We lose all the time and energy that went into growing those plants as well as lose all the revenue they would have brought in for us. Further, I love the urgency that combining data and the school garden creates because many students have learned the valuable life lesson that not making a decision is making a decision. If students freeze or do nothing when confronted with the data about the garden, that too has consequences.

> “ FAILURE TO INTERPRET DATA THEN ACT ON THEIR INTERPRETATION HAS REAL CONSEQUENCES: OUR PRODUCE COULD DIE

Using data to spot trends and make predictions

The other major way we use data in FH Grows is to spot trends and make predictions. Different to using data to create the ideal growing conditions in our garden every day, the sensors that we use also provide a way for us to use information about the past to predict the future. FH Grows has about two years' worth of weather data from our Raspberry Pi weather station (there are guides online if you wish to build a weather station of your own). Using weather data year over year, we can start to determine important events like when it is best to plant our veggies in our garden.

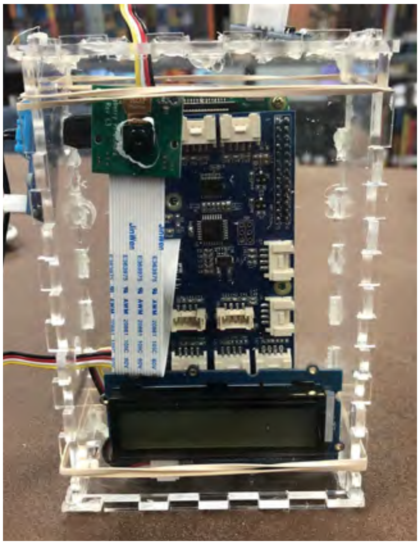
For example, one of the most useful data points on the Raspberry Pi weather station is the ground temperature sensor.



Last semester, we wanted to squeeze in a cool weather grow in our garden. This post-winter grow can be done between March and June if you time it right. Getting an extra growing cycle from our garden is incredibly valuable, not only to FH Grows as business (since we would be growing more produce to turn around and sell) but as a way to get an additional learning cycle out of the garden.

So, using two seasons' worth of ground temperature data, we set out to predict when the ground in our garden would be cool enough to do this cool veggie grow. Students looked at the data we had from our weather station and compared it to different websites that predicted the last frost of the season in our area. We found that the ground right outside our door warmed up two weeks earlier than the more general prediction given by websites. WIth this information we were able to get a full cool crop grow at a time where our garden used to lay dormant.

We also used our Raspberry Pi to help us predict whether or not it was going to rain over the weekend. Using a Raspberry Pi connect to Weather Underground and previous years' data, if we believed it would not rain over the weekend we would water our gardens on Friday. If it looked like rain over the weekend, we let Mother Nature water our garden for us. Our prediction using the Pi and previous data was more accurate for our immediate area than compared to the more general weather reports you would get on the radio or an app, since those considered a much larger area when making their prediction.



Our Pi prediction system has given us far more accurate data

It seems like we are going to be collecting even more data in the future, not less. It is important that we get our students comfortable working with data. The school garden supported by Raspberry Pi's amazing ability to collect data is a boon for any teacher who wants to help students learn how to interpret data and turn it into action. (HW)

CHRIS AVILES

Chris Aviles is a Raspberry Pi-certified educator and teacher at Knollwood Middle School in the Fair Haven school district in Fair Haven, New Jersey. There, he runs the renowned Fair Haven Innovates program he created in 2015.

Pupils creatively design, craft, program, and build interactive objects using Arduino with preassembled sensors and actuators, and Snap4Arduino

BRINGING PHYSICAL COMPUTING TO THE CLASSROOM

Why and how should we teach physical computing in computer science education? Here are some useful guidelines for lesson planning, and how to apply them in class

Physical computing is an interdisciplinary field, involving the creative arts and design processes. It brings together hardware and software components, joining up the virtual world of computers to the physical world of humans, all by using embedded systems design concepts (and neighbouring disciplines).

Products of physical computing make use of sensors and actuators to interact with their environment. Tools include microcontrollers and mini computers, often with extensions to facilitate component handling. Projects are of an iterative nature, quickly leading to working prototypes, such as the interactive garden at the end of this article. Ideas and intended interactions with the audience or environment are in focus when planning and creating interactive objects. Purposeful tinkering is encouraged to develop ideas and figure out how things work.

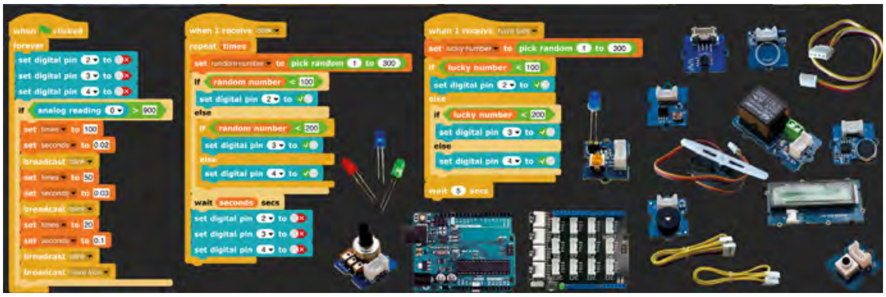
Benefits of physical computing

It all makes physical computing a promising approach for introducing embedded systems and underlying concepts in computer science (CS) teaching at secondary level. Many skills and competencies are gained. Programming concepts and control structures such as decisions, loops, variables, comparisons or arithmetic operations are used to make objects that can flash lights, move, or make sounds reacting to their environment. Content related to embedded systems design such as sensors, measurement, control or common practices when working in larger projects is relevant, too.

In physical computing, pupils learn with (and about) interactive objects and systems by creating tangible real-world products from their imagination. This can be used to promote creative learning in CS education and boost their motivation. It matches with the learning theory of constructionism. In this understanding, learning is most effective in contexts where learners construct knowledge and develop competencies from their initiative and for a personally relevant purpose, whilst engaged in creating visible

EXEMPLARY LESSON STRUCTURE

- Introduction and motivation: Provide examples: short tutorial session
- Tinkering: Let learners explore the tools, provide manuals and cheat sheets
- Brainstorming: Find ideas for projects
- Project planning: Give guidance with worksheets
- Presentation and discussion: Reflect ideas and plans
- Creation: Let learners work in groups of two to four
- Exhibition and reflection: Present the project to an audience (open-door day, school party…)



```
when [flag] clicked
forever
  set digital pin 2 to ✕
  set digital pin 3 to ✕
  set digital pin 4 to ✕
  if analog reading 0 > 900
    set [illegible] to 100
    set seconds to 0.02
    broadcast [illegible]
    set [illegible] to 50
    set seconds to 0.03
    broadcast [illegible]
    set [illegible] to 20
    set seconds to 0.1
    broadcast [illegible]
    broadcast [illegible]
```

```
when I receive [illegible]
repeat times
  set randomnumber to pick random 1 to 300
  if random number < 100
    set digital pin 2 to ✓
  else
    if random number < 200
      set digital pin 3 to ✓
    else
      set digital pin 4 to ✓
  wait seconds secs
  set digital pin 2 to ✕
  set digital pin 3 to ✕
  set digital pin 4 to ✕
```

```
when I receive [illegible]
set lucky number to pick random 1 to 300
if lucky number < 100
  set digital pin 2 to ✓
else
  if lucky number < 200
    set digital pin 3 to ✓
  else
    set digital pin 4 to ✓
wait 5 secs
```

MAREEN PRZYBYLLA

Mareen is research assistant in Didactics of Computer Science at the University of Potsdam. Her research focuses on physical computing in computer science education. She frequently teaches computer science courses at high-school level.

mareen-przybylla.de, **@MPrzybylla**

artifacts. With physical computing, these are not only visible but also tangible – similar to artistic sculptures.

Guidelines for teaching

To make it easier to start teaching physical computing in CS, guidelines and design principles for teaching and learning scenarios, as well as classroom-ready tools and materials, were developed.

Using a design-based research approach, theoretically-developed concepts were tested in classrooms and evaluated with pupils and teachers. Promoting constructionist and creative activities, such as tinkering and prototyping, has proven to be a key part of successful learning scenarios. Contrary to expectations, it is not necessary to strictly focus on project planning before introducing tools for positive impact in the evaluated domains, but it helps stimulate ideas and creativity. A commonality in less successful courses was missing structure in project planning and implementations. Providing scaffolds is particularly helpful in groups that are not used to project work.

This results in the following design principles for physical computing teaching:

1 Integrate tinkering activities in dedicated learning phases in which content knowledge and skills are acquired
2 Let learners create their own interactive objects ('pottery making approach')
3 Let learners develop working prototypes
4 Provide interesting themes: open topics that trigger imagination and creativity
5 Integrate creative methods
6 Integrate technical aspects with art/crafting
7 Provide scaffolds to structure the process of project work:
   a Planning from user perspective
   b Planning from developer perspective (non-technical and technical viewpoint)
8 Choose suitable construction kits and programming environments for the target group (low floors, wide walls, high ceilings)
9 Provide suitable crafting material and tools for the intended projects
10 Prepare a joint exhibition of all interactive objects
11 Present the results to an audience

> ❝ PROVIDING SCAFFOLDS IS HELPFUL IN GROUPS NOT USED TO PROJECT WORK

My Interactive Garden

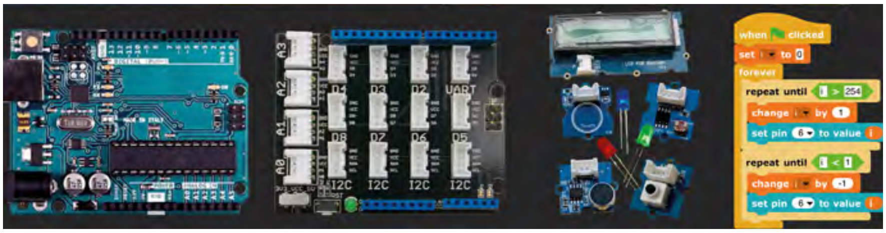
'My Interactive Garden' (MyIG) is a lesson series using these ideas. A theme of creating an interactive garden exhibition was used to encourage a variety of projects, and also call for ideas and collaboration. Learners design, craft, program, and build objects using a construction kit based on the microcontroller platform, Arduino, with preassembled sensors and actuators, and block-based programming environment, Snap4Arduino.

They are provided with material and worksheets that structure the project work. In tinkering activities and learning phases, pupils learn and acquire specific concepts and skills. For instance, they are introduced to sensors and actuators as a means of analog and digital inputs and outputs, to ideas of events that take place in parallel or serial, and to continuous-time and discrete systems.

During project work, learners develop prototypes in short iterations, which are discussed with their classmates and teacher throughout the process. They also detect problems and possible misunderstandings early and find solutions to occurring problems. Finally, the students present their projects during an exhibition and discuss their experience with the audience. They explain the purpose and functionality of their interactive objects and reflect their progress.

MyIG and other physical computing settings were looked at in a cross-sectional, pre-post intervention study. Courses adhering to the design principles were more successful than most others. Physical computing can motivate learners more than many other activities in CS classrooms. MyIG implementations on average showed motivation values over twice as high as general physical computing activities, proving better for appealing to female students. In 71% of the analysed MyIG courses, the existing gap between boys' and girls' learner motivation in CS classes narrowed down.

The consistent implementation of design principles for physical computing teaching generally showed very positive results. In addition to higher learner motivation, most pupils liked the projects and had more fun, and felt more competent than in their prior lesson series. (HW)



```
when [flag] clicked
set i to 0
forever
  repeat until i > 254
    change i by 1
    set pin 6 to value i
  repeat until i < 1
    change i by -1
    set pin 6 to value i
```

A FREE ONLINE COURSE TO SUPPORT TEACHERS WITH CYBERSECURITY

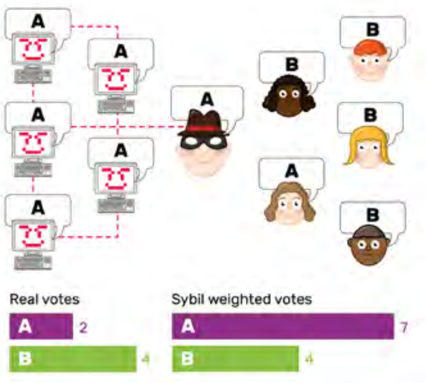
Develop your knowledge of cybersecurity, gain access to classroom activities, and take part in discussions worldwide

The Raspberry Pi Foundation has recently launched a free online course called 'Introduction to Cybersecurity for Teachers'. This course is aimed at teachers of GCSE Computer Science (ages 14–16), but is available to anyone who may want to learn about cybersecurity.

The course will run over three weeks, and involves around two hours per week during a time that suits you. There will be a number of activities to take part in in order to support your learning, such as joining in with discussions and coming up with role-playing scenarios for the classroom, and you'll be learning alongside people from all over the world. We'll also suggest several teaching activities, and you can download the videos, article text, and activity PDFs from this course to use in your own lessons.

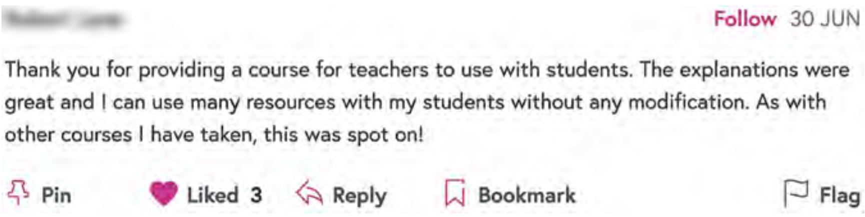
Cyber-attacks

Introduction to Cybersecurity for Teachers explores the core ideas of cybersecurity that should be taught in the classroom. Teaching about the different threats that people and devices are vulnerable to, and examining the steps that can be taken to keep data secure, is an important way to help students stay safe while using technology. Some of the threats that will form part of your discussions include attacks, such as data theft and corruption, as well as accidental breaches of data.



One type of Sybil attack can be used to swing votes on an e-voting platform

One of the most prevalent forms of attacks is social engineering attacks, in which victims are deceived into sharing valuable personal data. This part of the course begins with an overview of automated social engineering, before delving into phishing, pharming, and name generator attacks. Subsequently, you will find out about interactive social engineering that requires the attacker to interact with

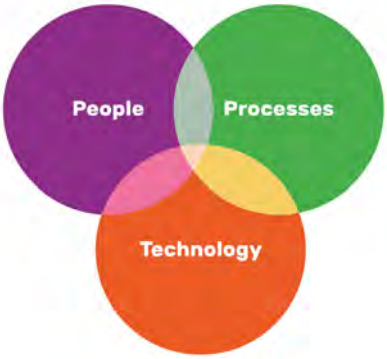


[redacted] Follow 30 JUN

Thank you for providing a course for teachers to use with students. The explanations were great and I can use many resources with my students without any modification. As with other courses I have taken, this was spot on!

Comment from a participant on completing the first run of the course

Cybersecurity approaches should utilise the three pillars of cybersecurity: people, processes, and technology



COMPUTER SCIENCE ACCELERATOR PROGRAMME

This course is part of the Computer Science Accelerator Programme, which is designed to support teachers in improving their knowledge of computer science and teach it at GCSE level. If you are a GCSE Computer Science teacher at a school in England, you are eligible for a free upgrade of the course to gain unlimited access: **teachcomputing.org/courses**. This course counts towards a nationally recognised qualification, 'The National Centre of Computing Education certificate in GCSE Computer Science subject knowledge'.

the victim more personally. The two types of attacks explored here are blagging and shouldering, which culminates in suggestions of classroom exercises.

Defensive measures

In this course you will learn about good cybersecurity practice, which utilises the three pillars of cybersecurity. Building on the strong foundations of these pillars enables individuals and companies to form robust preventive measures against exploitation by malware, malicious bots, SQL injections, and physical threats. You will also build your knowledge of the different tools that protect data and websites. These include strong passwords, biometrics, two-factor authentication, and firewalls.

Another important aspect is the legal solutions available in the UK to protect and prosecute. Discussions will revolve around new laws that have been created in an attempt to deal with cyber-attacks; how effective are these and what rights do you have to access your data held by an organisation?

> " ANOTHER IMPORTANT ASPECT IS THE LEGAL SOLUTIONS AVAILABLE IN THE UK TO PROTECT AND PROSECUTE

From spyware to Sybil attacks, physical security to penetration testing; you'll be learning about a range of topics in an increasingly relevant part of the computer science syllabus and our daily lives.

Sign-up for the Introduction to Cybersecurity for Teachers course is available now by heading to the website **rpf.io/cybersecurity**.

More information about this and other courses that are part of the Computer Science Accelerator Programme can be found at **teachcomputing.org/courses**. (HW)

ALEX PARRY

Alex is a Learning Manager at the Raspberry Pi Foundation and Computer Science teacher. When not coding, he enjoys table tennis, gaming, and wandering in the woods.

A YEAR WITH DESIGN JOURNALS

Introducing design journals for Key Stage 2 pupils – and how it went…

The national curriculum for computing in England states that computing has deep links with science, and design and technology. In my school, in both these subjects, pupils are given exercise books in which they are required to make plans and do designs. The national curriculum also states that pupils in Key Stage 2 should be able to design and write programs.

With this in mind, I decided to make sure that pupils spent time in the design stage of their computing projects and that I would facilitate this with the introduction of specific design journals.

So, last September I introduced a computing design journal which was simply an A4 book with plain pages. I decided on plain pages as I didn't want the pupils to feel constrained but to have the freedom to develop their designs in any way they saw fit.

This article explains how we have been using design journals and presents responses and views from the pupils' perspective. It covers three main aspects. Firstly, how the journals have helped children to better understand their computing projects. Secondly, how the journal has helped them to self-assess their work and indicate their levels of confidence. Lastly, to help pupils improve their computational thinking and problem-solving skills.

Recent research papers have suggested that design can help novice and struggling programmers develop an awareness of what is doable, and also that design helps with self-regulation much in the same way as planning does in writing. The research is outlined in an article called 'Abstraction in action: K-5 teachers' uses of levels of abstraction, particularly the design level, in teaching programming', and the piece is available to read free of charge in the International Journal of Computer Science Education in Schools.

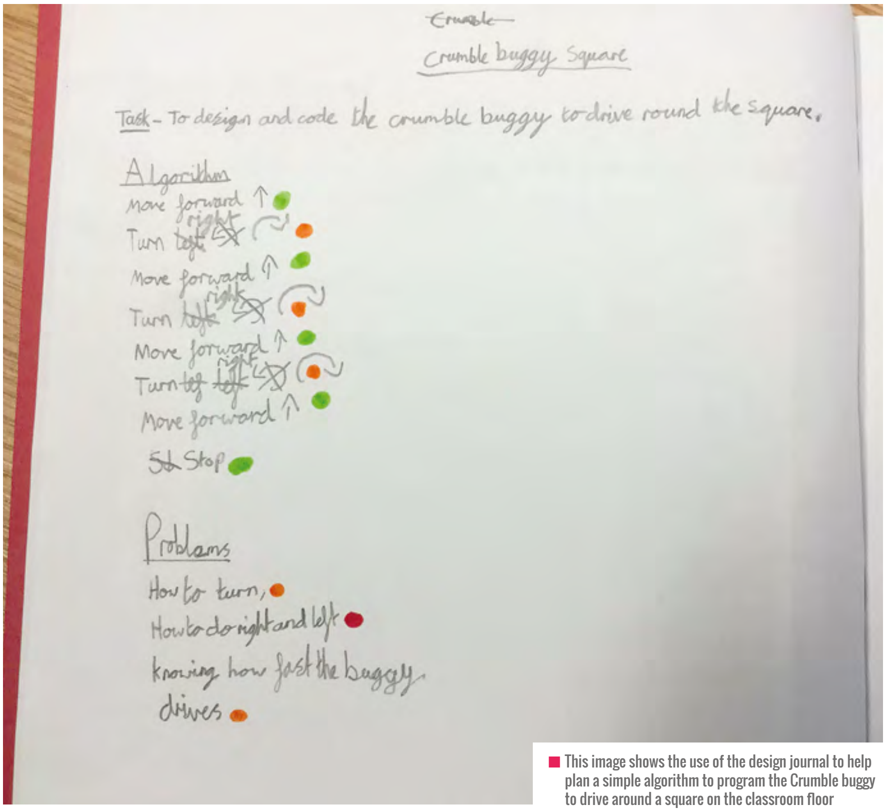
The initial reaction

I decided to give design journals to pupils in Years 4 to 6 in my school. The initial reaction from the pupils was one of shock and delight. Many of them opened their books and questioned why they had blank pages. My reply was that "These are not exercise books like you get in English, but these are design journals – they are a place where you can formulate your ideas and solve problems."

The children took my word for it and were excited by the idea and intrigued by how they could use them throughout the year.

Design in computing lessons

I 'sold' the journals to pupils as being different to exercise books. I explained that they were more like the sketchbooks they used in art or design. I didn't constrain the use of the journals but wanted the children to engage with them and get to grips with the design process in computing.



~~Crumble~~

Crumble buggy Square

Task – To design and code the crumble buggy to drive round the square.

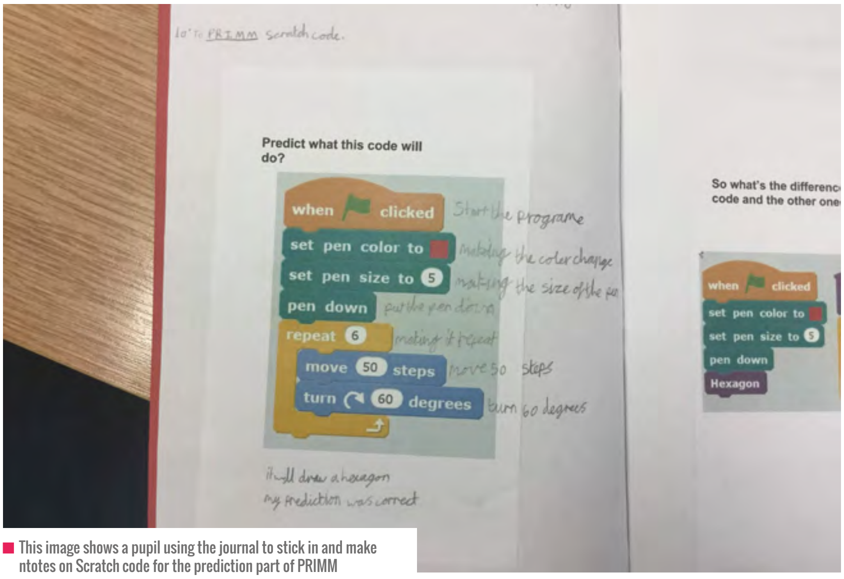
Algorithm

Move forward ↑ 🟢
Turn ~~left~~ right 🟠
Move forward ↑ 🟢
Turn ~~left~~ right 🟠
Move forward ↑ 🟢
Turn ~~lef~~ ~~left~~ right 🟠
Move forward ↑ 🟢
~~St~~ Stop 🟢

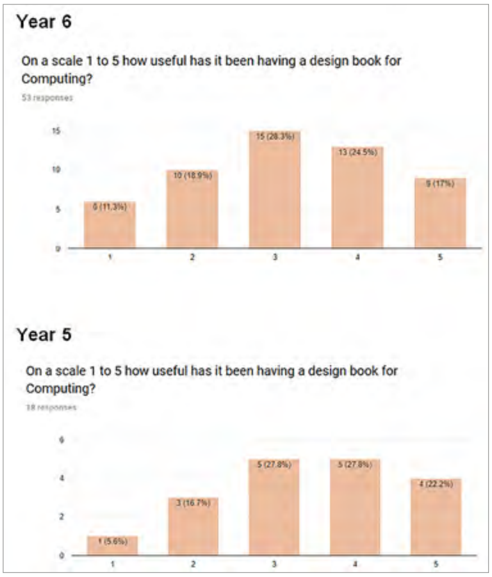
Problems

How to turn, 🟠
How to do right and left 🔴
knowing how fast the buggy drives 🟠

This image shows the use of the design journal to help plan a simple algorithm to program the Crumble buggy to drive around a square on the classroom floor


lo: To PRIMM Scratch code.
Predict what this code will do?
when clicked Start the programe
set pen color to Making the coler change
set pen size to 5 making the size of the pen
pen down put the pen down
repeat 6 making it repeat
move 50 steps move 50 steps
turn 60 degrees turn 60 degrees
it will draw a hexagon
my prediction was correct
So what's the differenc
code and the other one
when clicked
set pen color to
set pen size to 5
pen down
Hexagon

This image shows a pupil using the journal to stick in and make ntotes on Scratch code for the prediction part of PRIMM



Bar charts showing responses to having a design journal in computing. The scale is 1 to 5, with 1 being 'not useful' and 5 being 'extremely useful'

So, whenever we did a computing project throughout the year, I would ask the children to use the journal to plan their ideas, storyboards, and algorithms, encouraging them to make informal notes and sketches with anything that they thought might help them.

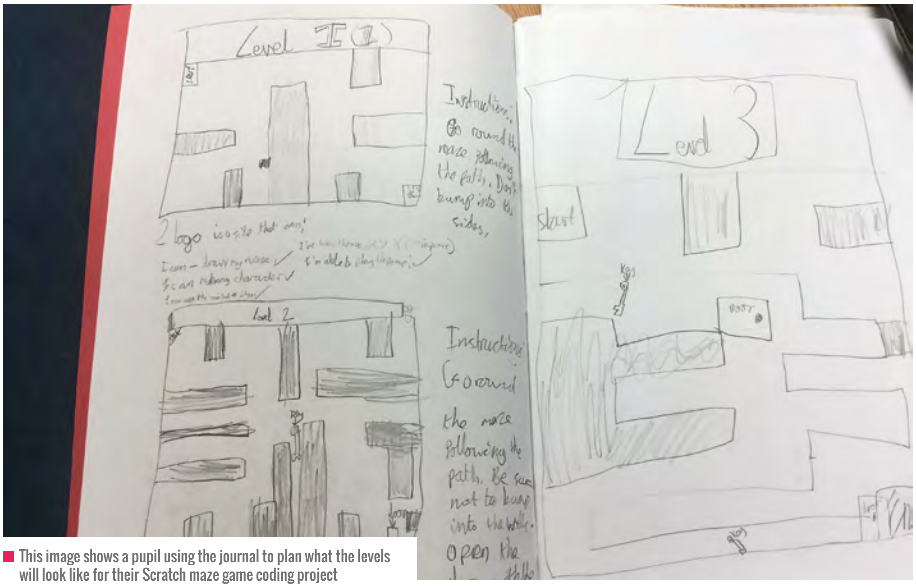
The journal was also a useful place for them to stick sheets in, for example, during the prediction part of PRIMM where they could write notes all around the sample code and self-assess their responses using traffic light colours. They could use it if they had to figure out a problem, or design an algorithm, or look at what the levels would look like in a scratch game.

It's fair to say that I'm still experimenting with what works and how these books fit into the workflow of the lessons. The images alongside this article highlight the versatility of the design journal and the reason I went for an A4 book with blank pages.

Pupil voice

In the summer term of 2019, I wanted to assess the impact that the books had on pupils' learning of computing, so I

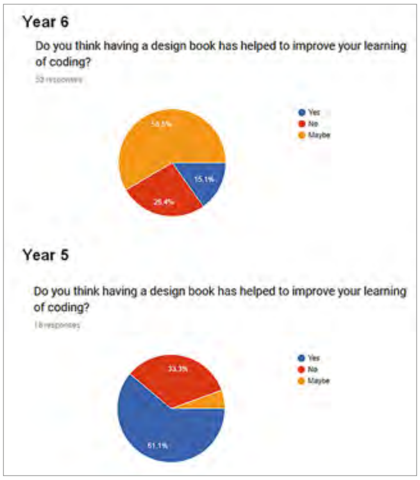
> I DIDN'T WANT TO CONSTRAIN THE USE OF THE JOURNALS, BUT WANTED THE CHILDREN TO ENGAGE WITH THEM


Level 1
Instructions: Go round the maze following the path. Don't bump into the sides.
Level 2
Instructions Go forward the maze following the path. Be sure not to bump into the walls. Open the
Level 3
Start

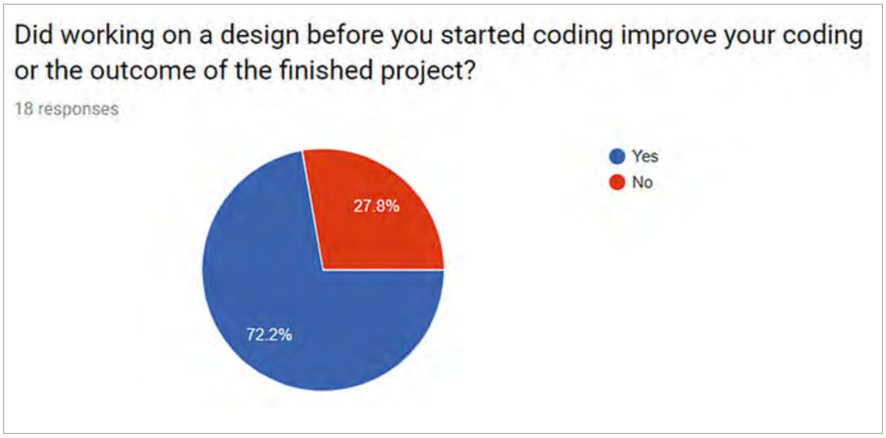
This image shows a pupil using the journal to plan what the levels will look like for their Scratch maze game coding project

MATTHEW WIMPENNY-SMITH
Matthew Wimpenny-Smith is leader of Digital Strategy and Computing subject leader, working for Headington School Oxford for the last seven years in the Prep School teaching EYFS, KS1, and KS2. On top of that, he is CAS Master Teacher and Oxfordshire Primary Community Leader. Also BCS Certified Computer Science teacher, Raspberry Pi educator, Google L1 and NCCE facilitator and generally passionate about all things computing/education.

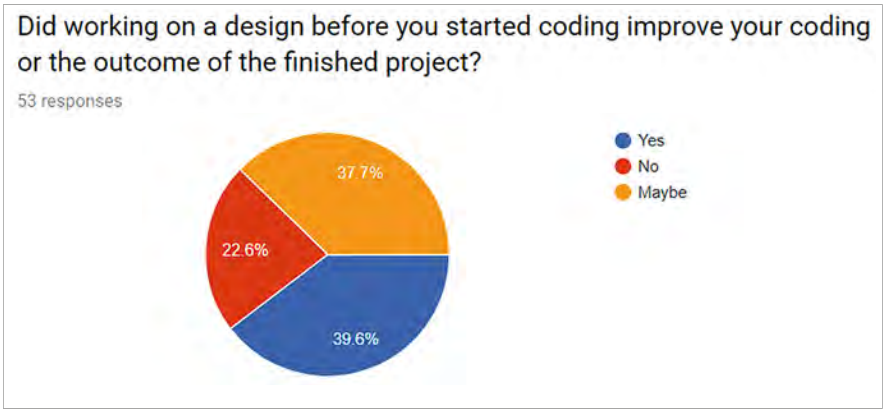
JANE WAITE
Jane Waite works for Queen Mary University of London. She undertakes research in computer science education and provides teacher training and support for resource development. Jane worked in the IT industry for 20 years and was a primary school teacher for ten years.



Pie charts showing impact of the journals on improving student learning of coding



Pie chart showing Year 5 responses to a question on the impact of a design-first approach on coding



Pie chart showing Year 6 responses to a question on the impact of a design-first approach to coding

> HOWEVER, THIS IS EARLY DAYS, AND I DID INTENTIONALLY KEEP THE ROLE OF THE JOURNAL LOOSE TO ENABLE CREATIVITY

conducted a pupil voice survey with Year 5 and 6 pupils. I asked them a range of questions, and I found some interesting results. I should note that I got more responses from Year 6 (53) than I did from Year 5 (18), due to other events taking place in school at the same time as the survey.

Still, the children found having a design book more useful than not for computing, and this was reflected both within the written responses of the pupils, which generally showed a positive impact on learning. It is notable that nearly half of both Year 5 and 6 found it useful or extremely useful. One Year 6 pupil commented that “it is much easier to then actually do the task because you have already planned it out! You’re more sure of what to actually do. I would definitely say to use the design books again!”

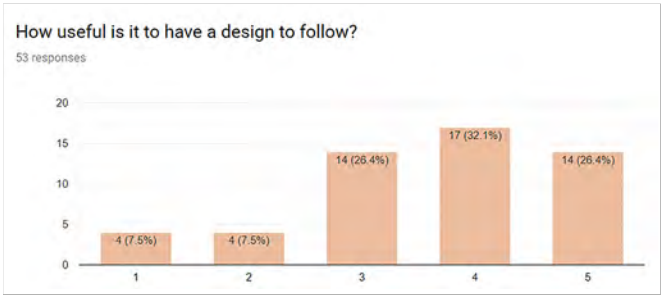
To see all the written responses from the survey please follow this link: **bit.ly/2krBgoN**.

Despite nearly half the Year 5 and 6 pupils thinking that the design journals were either very or extremely useful, they weren’t so sure it helped with their actual coding. In fact, over half of the Year 6 respondents said “Maybe” when asked whether they felt having a design book improved their learning of coding. I wonder, though, if the question was a little muddled.

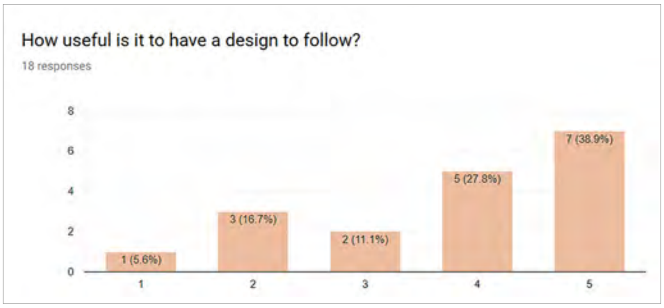
Although the sample size was a lot smaller, the Year 5 response to the same question showed that well over half of the pupils were more positive when asked about improved coding from a design-first approach. Still, just shy of a third of pupils in Year 5 and 6 answered no, that it didn’t help their coding. I would say that at this age group, there are always a number of pupils who don’t see the benefit of doing designs or planning before getting started, independent of subject.

On reflection, I should not have included ‘maybe’ as an option, or perhaps I could have included a scale similar to the usefulness of journal question. Also, it was not clear as to whether all the students associated design with improvements in coding. Next term, I will think about how I define the role of the design journal better so it has more purpose in the lessons. However, this is early days and I did intentionally keep the role of the journal loose to enable creativity. I wanted to facilitate the process of design and encourage learners (and myself) to experiment and to see how the journal might fit into the design activity, rather than have the journal seen as a product in itself.

In another question to pupils. I asked about the impact of design, and whether they felt it helped improve the outcome of their coding project. I was keen to investigate students’ views on the impact of designing before coding, and whether they felt this approach improved the outcomes of the finished project. Both year groups were positive on this: i.e. that having a design before starting the coding



Bar chart showing Year 6 responses to a design-first approach. The scale is 1 to 5. with 1 being 'not useful' and 5 being 'extremely useful'



Bar chart showing Year 5 responses to a design-first approach. The scale is 1 to 5, with 1 being 'not useful' and 5 being 'extremely useful'

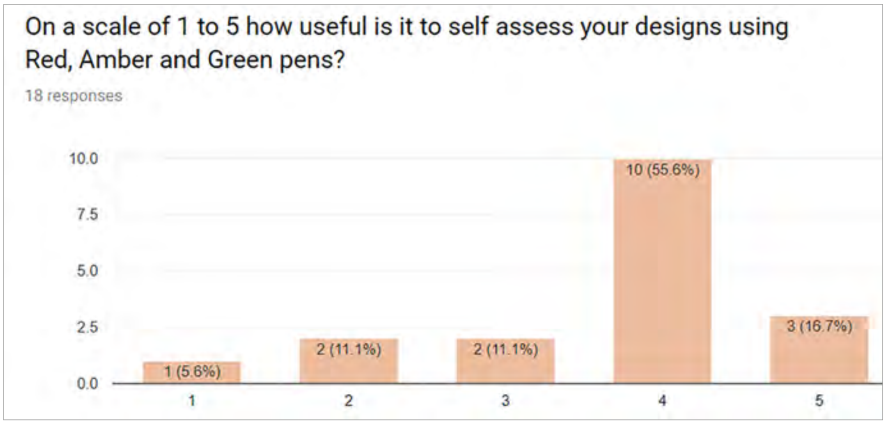
© Images courtesy of Matthew Wimpenny-Smith

improved the outcomes of the project. Again, perhaps I shouldn't have offered 'maybe' as an option here. I now need to develop a way to measure the impact of a design journal perhaps with a group who don't do design versus a group who do.

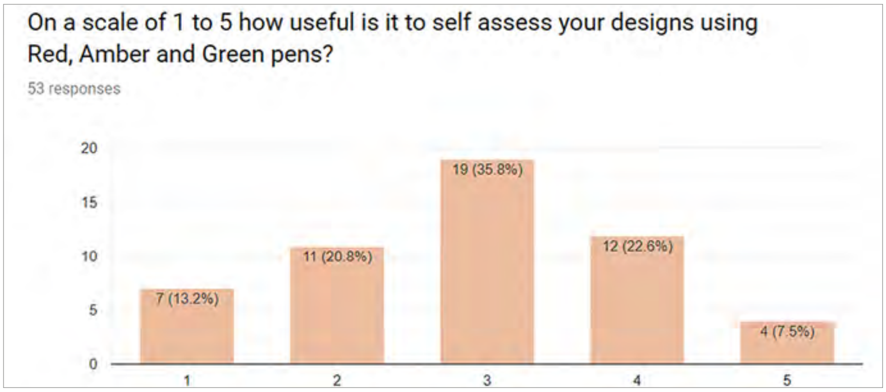
Positive responses to a design-first approach were repeated in responses to a second question on how useful it was to have a design to follow in the first place. Interestingly though, the Year 5 pupils found it more useful than Year 6. This may have been due to the unmatched sample size between the two cohorts, or it may be because of different material being covered. Or maybe something else! I will have to compare next year and perhaps ask some open questions, or ask pupils to discuss it all between them and make notes on their discussion.

During the design stage of projects, I would often ask the pupils to assess their designs for confidence using the red, amber, and green traffic light colours, along with 'purple polishing pen' annotations and edits similar to the 'purple editing pen' that they were used to using in their literacy lessons. Using this method helped the children gauge an understanding of what they considered to be doable. This continual annotation helped me to identify gaps in their understanding but also helped to make their computational thinking and problem-solving more visible. It also acted as an indicator to me (and to them) of how confident they were feeling about their programming projects as they were being developed.

The responses to the survey found that over 60 percent of all the pupils surveyed found this method of self-assessment useful or very useful. This has highlighted to me that this is an effective way for pupils to gauge their own confidence and understanding of a project and its design, and is an effective tool for AfL (Assessment for Learning).



Bar chart showing Year 5 responses to impact of traffic light self-assessment. The scale is 1 to 5 - 1 is 'not useful' and 5 is 'extremely useful'



Bar chart showing Year 6 responses to impact of traffic light self-assessment. The scale is 1 to 5 - 1 is 'not useful' and 5 is 'extremely useful'

Conclusion: Stick or ditch design journals?

Based on the responses to the pupil voice survey and my own experience, I definitely intend to use design journals for Key Stage 2 in the new academic year. On reflection, I need to develop my practice to incorporate them more in my teaching, and to develop their purpose and role as a tool to build their design-first approach. Also, to improve the visibility of computational thinking and problem-solving.

It's clear from the experiment this year that the children benefitted from having a journal in the subject at this level. I also need to be consistent with what I call them – either a design journal or a design book. I think defining them as a journal makes them more of a 'creative' and 'process' space, whereas a book is seen more as an end product, an outcome space for polished work requiring marking and scrutiny. I might just go with journal... (HW)

Mila F has been researching near-death experiences, and she is testing out her Cospaces creation on the Oculus Go

EMBRACING VR IN THE CLASSROOM

Reimagined landscapes, near-death experiences, and room design are just some of the possibilities when engaging students with designing in virtual reality

Giving students access to more tools can lead to some amazing creations. At University Liggett School in Grosse Pointe Woods, MI, grade 12 students are given the opportunity to explore areas of interest to them and share their learning at the end of the school year. The Action Research Program has become a staple of our educational program, and students have come up with some amazing projects. One student is exploring the design of origami heart stents, another is investigating facial reconstruction in Forensic Science, and others are exploring topics that require virtual reality to fully immerse their audience in a learning experience. The thought of designing in VR might sound like a scary endeavour, but it has been made much more accessible thanks to Cospaces.io.

What is Cospaces?

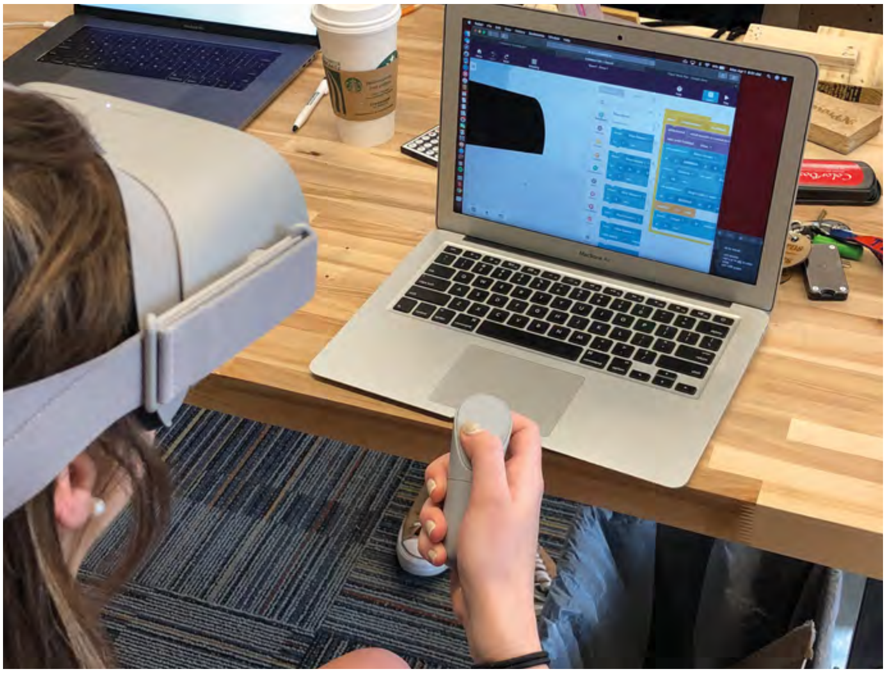
Cospaces is a web-based platform that allows users to create in VR. It has a free and a paid model that are options for teachers and students. The free version is perfect for any student or teacher to explore what the program is like and how it might be used. It's easy to create your own account and, once you've registered, you can create your own worlds, explore worlds others have created, create classes, invite students, and so much more. I recommend clicking on the My Spaces tab on the left, and going through the Tutorial world they created for you.

Once you have gone through the tutorial, you will notice that Cospaces works with a click-and-drag approach to creating. A user selects a background and starts dragging characters, buildings, and other items onto the work plane. Once everything is on the work plane that is needed, the next step of interactivity allows the users to code interactions using a block-based coding system. It is a great way to add an element of interactivity to the VR experience.

Students' use of VR

Some students will have projects that require an element of immersion that can be hard to replicate in any other way. As teachers, we are always on the lookout to find tools that help engage students in specific content. Sometimes, we need to find tools that allow students to engage in the content that is of interest to them. That freedom to explore topics that are meaningful to them increases engagement tenfold. Here are three examples of how students are using VR and Cospaces to explore areas of interest to them.

Student Mila F is exploring what people see and hear as part of reported near-death experiences. She dove into the coding aspect of Cospaces and was able to recreate what a near-death experience looked like, based on her research. She even figured out how to embed a YouTube video in the experience to create a sense of motion.



NICHOLAS PROVENZANO

(@thenerdyteacher) is Makerspace Director and Technology Coordinator for University Liggett School. Raspberry Pi Certified Educator and author of Your Starter Guide to Makerspaces and The Maker Mentality.

COSPACES UPGRADE COSTS/BENEFITS

5 seats - 75 USD per year	20 seats - 120 USD per year	30 seats - 180 USD per year

Premium Upgrades:

Unlimited Object Library
Unlimited World Creation
Unlimited External Imports
Expanded Code Block Library
Unlimited Number of Classes
Unlimited Number of Assignments

> THESE THREE STUDENTS ARE USING VIRTUAL REALITY IN VERY DIFFERENT WAYS AND THAT IS THE BEAUTY OF IT

Another student, Alex A, is creating a VR model of a part of a city, as it could look if there was a renewed investment into a specific community. He is designing buildings and streets to give a person an opportunity to walk through the new VR city he has created.

Madison B is creating rooms based on her research on feng-shui. Her virtual environments are designed to test how people feel based on specific design elements that are supposed to impact people in various ways.

These three students are using virtual reality in very different ways and that is the beauty of it. Virtual reality platforms, like Cospaces, are a blank canvas that allows the user to create whatever they want. The virtual sky is the limit!

Why use VR?

Some of you must be thinking, 'this is nice, but why should I start exploring virtual reality with my classes'? This is wonderful question and the answer is pretty simple: students are going to enter a world where virtual reality is commonplace. Technology is advancing at such a pace, that many jobs will require some aspect of VR designing or interaction. Engineers and industrial designers are already exploring how VR can be used as part of their work. It is important that teachers continue to push the envelope of technology use, because our students will need these experiences to prepare them for the world ahead of them, not the one behind them.

I am working with a Spanish language teacher who is working to create a virtual environment where students will engage with virtual characters and understand the directions they are given in Spanish. Instead of creating everything from scratch, she is using the party template provided by Cospaces, and just changing the word bubbles from English phrases to Spanish ones. As she becomes more comfortable using Cospaces, she will be able to record audio and embed more content so students can immerse themselves more fully in the world she creates for them.

It always takes a leap of faith when deciding to use any new piece of technology and, as teachers, it is our job to take those leaps and support students as best as we can. Virtual reality is not going to fit for all of your lessons. It might fit only one and that is fine. Giving students just a taste of this new software can be the thing that creates a lifelong love of virtual design. (HW)

CODERDOJO AND TECHGIRLZ: FUN IN THE CHALLENGE

Kids learn more when they don't follow directions – spice up your activities with more freeform challenges

Since retiring, I've been working with kids ranging from 7-17 at my CoderDojo RTP and with middle school girls through TechGirlz. With such a wide range of ages and skills, one of my main roles is to suggest projects for the kids that keep them interested and coming back for more. Fortunately for me, over the last three years, a lot more projects specifically targeted at kids have been created. My favourites to pick and choose from are CoderDojo, Code Club, Raspberry PI, and code.org.

I typically start all first-timers (especially those under 12) with Scratch. The drag and drop interface is great for young kids without strong typing skills. It also prevents early frustrations from those getting started with Scratch, and prevents typos, syntax errors, and such-like.

I start with the Scratch tutorials and CoderDojo beginner sushi cards, and then move to Code Club projects that are now organised into progressively harder modules. When the kids are ready, I give them Scratch challenges, to help determine if they are ready to try Python or HTML. Girls in particular enjoy the website activities, and many start doing their school projects using HTML. For the really young kids, I prefer to keep them on drag and drop programming experiences and use micro:bits and Dash & Dot robots to mix things up. Don't worry if kids try Python or HTML and go back to Scratch. Even the most talented kids usually return to Scratch for a while – and their advanced projects provide inspiration for the younger kids.

Nothing beats a good challenge

While all the step-by-step projects are great and the kids can do them without much help, are the concepts really sinking in? With some, it's obvious they are doing more than blindly following the directions. Some of the kids never finish a project as written, as they find something cool and away they go in their own direction. However, others may need a little more structure and a bit of a push to get their creativity really rolling. For both sets of kids, I've been very successful with 'challenges'.

> SOME OF THE KIDS NEVER FINISH A PROJECT AS WRITTEN, AS THEY FIND SOMETHING COOL AND GO IN THEIR OWN DIRECTION

My Scratch challenges are freeform: a program to build or a feature to add but without the support of step-by-step directions. Kids really get a chance to apply what they've been learning. My favourite challenge is still the first one we came up with: 'Tucker's coin toss challenge'. With a blank canvas, can they meet all the challenge requirements? Did they need help? Do they surprise me with their creativity?

Completing the challenge requires most of the basic Scratch concepts, and is a good indicator of whether a kid is ready to move off Scratch.

'Ruth's traffic light challenge' is another fun Scratch challenge using the Raspberry Pi. This challenge requires wiring green, yellow, and red lights on the Pi, and programming them in Scratch. The kids make a Scratch car visualisation that drives according to the light. The first kid to complete the challenge also implemented a visualisation of the traffic light, so that's now part of the challenge, too. For extra fun, I tell the kids to make the car go faster on yellow. This challenge often provides some really funny demo bugs.

First timers on Python typically start with either the CoderDojo sushi cards or turtle races. At the end of the CoderDojo sushi cards is the specification for building a guessing game. The guessing game is quite a good challenge – not too much of a stretch for most kids, but keep an eye out as some do struggle. I also have some challenge extensions to make the guessing

KIDS LOVE SWAG

When my Dojo parents go to technology conferences, I ask them to bring back kid friendly swag. At the conclusion of each Dojo, we do an attendance drawing and the winner gets a pick from the swag bag. The kids love to watch the random number generator to see who wins, and the side benefit is everyone signs in.

RUTH WILLENBORG

Ruth Willenborg, a retired IBM Distinguished Engineer, is the founder and champion of IBM CoderDojo Research Triangle Park (RTP), NC and a Triangle Techgirlz volunteer instructor. You can contact Ruth at **coderdojortp@gmail.com**.

game more flexible which requires the kids to work with variables, add loops, and suchlike. We also have the 'M&M Challenge', which is the inverse of the guessing game and makes a great follow-on.

Both the guessing games and turtle races are great for demos because the whole club gets involved in playing them. Twenty kids cheering on different-coloured turtles gets pretty loud. Extensions to turtle races, to identify which turtle won and add a slow-motion instant replay, are my newest challenges, and also help avoid some serious demo controversy when the winner isn't obvious!

The more, the merrier

Finally, I'd like to touch on this. One of my issues is getting kids who come alone integrated and working with others. Retention is typically higher when kids come with friends and family members. I have a few ideas here, but I'd love to hear suggestions from you.

For the young kids, working together to program Dash & Dot is a good activity, and micro:bit projects with radio capabilities lead to good interaction between slightly older kids. For middle school-age kids, Raspberry Pi Parent Detector is a super fun project to do, paired up in a group setting. I've made some modifications to streamline and complete it within a 90-minute session. My modified flow gets the camera going first, which really gets the kids engaged, and by the end they are capturing and replaying movies based on the motion sensor.

Demo time is also a great opportunity for interaction. One of our Dojo's intermediate and advanced badge criteria includes incorporating one or more suggestions from club members. This gets kids really paying attention to the demos and yelling out suggested improvements. I have some kids who both suggest improvements and share the programming features that can be used to implement it.

You can find more information on the challenges, badge criteria, and other projects I've discussed throughout this feature at my website. Just go to: **coderdojortp.wordpress.com**. (HW)

TT, THE TALKING LINUX PENGUIN

TT is a great teaching aid for getting kids excited about interacting with technology. TT is the open source Linux-stuffed Penguin pattern, with an imbedded Raspberry Pi and the components of a Google AIY 1.0 voice kit. Kids program TT using Python and Google text to speech APIs. More on TT at: **coderdojo.com/2019/04/18/introducing-tt-the-talking-linux-penguin**

FOSTERING DATA LITERACY COMPETENCIES IN SCHOOL

How can larger amounts of data be analysed in schools? Why is this topic important for students? Current research on data literacy gives an answer to these questions

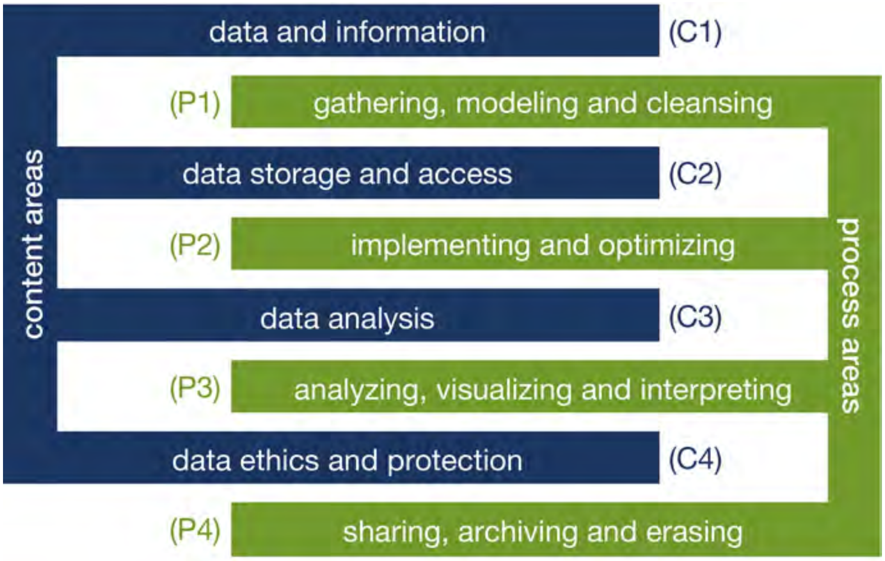
Today, data analyses are everywhere. For example, large companies collect masses of data and analyse it with the goal to systematically promote their products, whilst social networks use data analyses to suggest 'friends'.

Nowadays, everyone is confronted with new challenges due to the increasing and widespread relevance of data, for example the necessity to decide which personal and foreign data are shared with others (including services on the internet), under which conditions, and for which purpose. At the same time, the question arises what others can do with, and read from, this data. Accordingly, not only do students need to acquire skills in this area, we all need to become data literate.

Data literacy competencies

But what does data literacy mean? According to widely accepted definitions, data-literate people are able to work with and handle data in a meaningful way. For example by acquiring, structuring, or analysing it.

In recent years, we further investigated this topic from a computing education point of view, taking into account various perspectives. On the one hand, we regarded the technical perspective on the large topic data, and on the other hand we also considered the students' and teachers' perspectives, as well as requirements coming from society. On this basis, we developed a data literacy competency model.


content areas
data and information (C1)
(P1) gathering, modeling and cleansing
data storage and access (C2)
(P2) implementing and optimizing
data analysis (C3)
(P3) analyzing, visualizing and interpreting
data ethics and protection (C4)
(P4) sharing, archiving and erasing
process areas

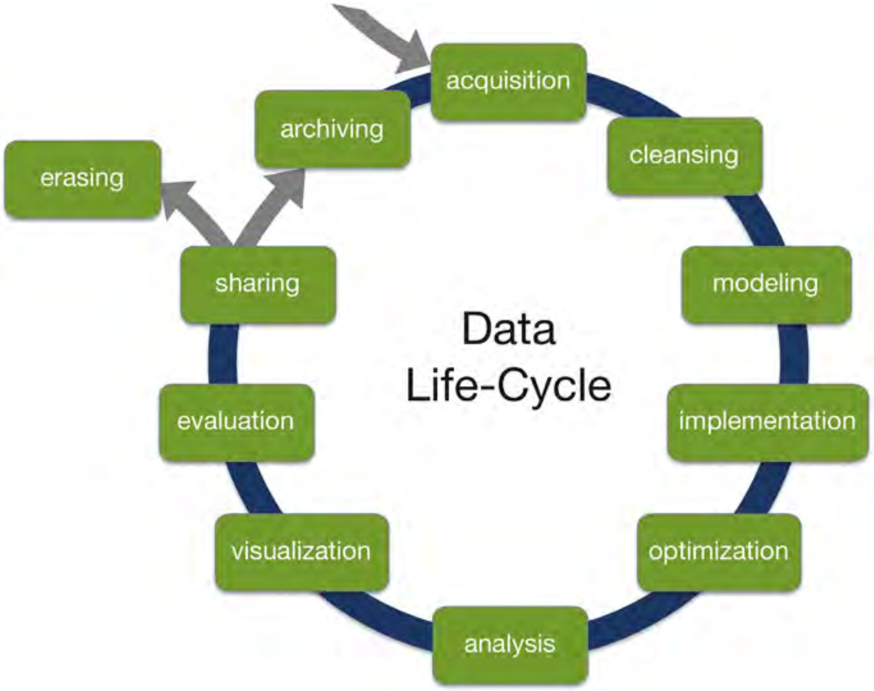
■ According to the research-based data literacy competency model, each data literacy competency must have both content-related and process-related aspects

This model characterises competencies related to data from two different perspectives resulting in two areas: the content areas clearly emphasise technical aspects, and hence are focused on the computer science content, while the process areas take a rather practically oriented perspective, and illustrate what can be done with data.

The two types of competency areas are closely intertwined, thus each data literacy competency has to connect to at least one content and one process area. For example, the competency to visualise data and analysis results incorporates both a content aspect (such as knowing different visualisation methods and their purpose) and a process aspect (covering being able to prepare data in a way suitable for visualising them and creating the aspired visualisation). Although the competency model was developed with a focus on computing education, due to its structure it can also be adapted to incorporate aspects from other subjects. After all, computer science is not the only subject that has to deal with data today. Other subjects can contribute important aspects too, particularly to the content areas, hence enriching the model and extending its usability beyond computing education.

The life cycle of data

When trying to include data literacy competencies in school teaching, often the question arises as to where to start. Most computing lesson plans likely have various connection points to data literacy,



The data life cycle model gives an orientation for data-literacy-oriented teaching, for both teachers and students

DR. ANDREAS GRILLENBERGER

Andreas Grillenberger is a research fellow at Freie Universität Berlin in Germany. Since 2013, he's worked on the topics of data, data management, and data literacy. He investigates these topics from a computing education perspective, particularly with focus on secondary education.

so there are various possibilities for data literacy teaching. Yet, it is important to keep in mind the whole process of working with data. When only discussing distinct parts of this topic, for example the analysis, other important aspects are missing (such as gathering data or justifying the analysis from an ethical perspective).

Hence, as a guideline for data literacy teaching, we developed the data life cycle model. This model gives teachers and students an orientation when working with data and sets its emphasis on the whole process of working with data, not just a small excerpt of it. Of course, not all aspects can be considered in the same depth in school, but using the data life cycle as an orientation helps to bring together all the knowledge and skills students acquire throughout computing education.

For example, aspects related to data modelling, implementation, and optimisation are typically already there in most computing curricula, and hence only need to be brought into connection with other aspects of the data life cycle. Also, when working with databases, real data could be acquired in class and structured for efficiency, storing them in the database instead of discussing rather fictitious examples. The important question, therefore, is not where to start with teaching data literacy, but where to connect it to what we teach already. The data life cycle helps to identify such connection points.

Fostering data literacy in school

When data literacy competencies are to be fostered in school, several challenges have to be overcome. Suitable tools have to be identified, appropriate examples that can motivate students and that concern them have to be selected, and concepts need to be worked out on how to foster these skills.

Particularly, as most data literacy competencies cannot be gained by theoretical considerations only, suitable examples and appropriate data play an important role for teaching data literacy. Such data may be acquired from various sources today. For example, programming interfaces (APIs) of widely known services on the internet (such as Twitter) aren't the only data sources that can be used. There are also rich and easy-to-use data sets that are released, such as by public administrations as part of open data projects (for example, open data can be found at **data.gov.uk**).

An exemplary project, particularly considering data analysis, is based on using real data about school students (e.g. a dataset about Portuguese students that was released on the UCI Machine Learning Repository, that can be found at **archive.ics.uci.edu/ml/datasets/Adult**), which can be analysed by students using simple tools (such as Orange, at **orange.biolab.si**), with the purpose to predict students' grades based on the information contained in the dataset.

As this setting directly concerns students, particularly if it is presented by the teacher as a possible new way to grade them, several ethical problems are raised that directly affect the students. The resulting discussions on challenges, risks, and opportunities that arise, along with the possibility to work with and analyse large amounts of data directly, lead to another challenge. Although this article has taken a computing education perspective on data literacy, this topic also affects and is relevant for many other subjects. Data literacy should therefore be considered an interdisciplinary topic and taught accordingly in school. Fostering data literacy in school is an open challenge to which we all can, and must, contribute in order to prepare our students for a life in a world where data is used continuously and everywhere. (HW)

RETRIEVAL PRACTICE

An effective learning strategy and an evidence-based teaching technique, **Gemma Moine** explains how she uses retrieval practice in her Computer Science classroom

In recent years, educational research has praised the findings of cognitive psychology research on retrieval practice, the idea that bringing information to mind can boost learning. It's been revealed that the mechanics of the memory has a large impact on learning. Understanding research and translating this into the classroom is key, but it can be very difficult for teachers to put into practice when workloads are high and time is precious.

The majority of retrieval tasks undertaken in my classroom have been taken ('magpied') from amazing teachers posting examples on social media and adapted for Computer Science. I highly recommend you research further into the topic using the original source links provided for further examples and ideas. This article aims to give teachers a toolbox of simple retrieval tasks that can easily be embedded into the classroom. Tasks are ordered in preparation time to help teachers decide which ones they may like to trial in the classroom.

Brain dumps

How: One of the easiest retrieval practice tasks you can incorporate into the classroom with minimal preparation are 'brain dumps'. Students have five minutes maximum to write down as much as they can recall on a specific topic given by the teacher. Answers can be written on paper or – my favourite – writing straight onto the desk with board markers. Students love a bit of desk graffiti.

Extend: Another useful task is for students to identify areas to revisit by highlighting missed areas on the brain dumps on their knowledge organisers or mind maps. This could be extended further to a homework activity where students are required to make flashcards on identified missing topics and then test one another at the beginning of the next lesson.

Original idea: Magpied and adapted from **@RetrieveLearn**.

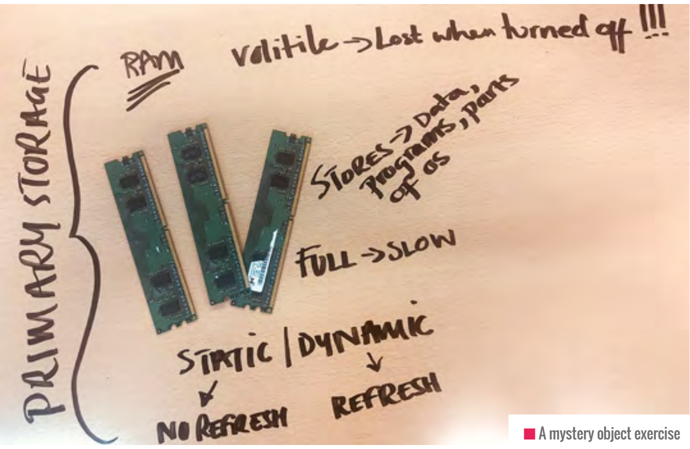
Take Three

How: A super-quick and easy-to-embed retrieval task is Take Three. Ask students to write down on paper three things they learned last lesson, last week, or last term. Give students the opportunity to think-pair-share what they have written down.

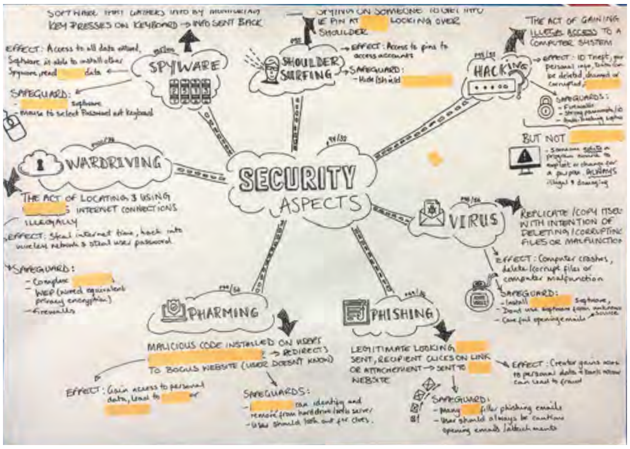
Extend: Ask students to find peers with written comments from the same topic (or where there are linkages between topics) and discuss.

Mystery object

How: Place an object on the desk and ask students to recall as much as they can


PRIMARY STORAGE
RAM
Volitile -> Lost when turned off !!!
Stores -> Data, Programs, parts of OS
FULL -> SLOW
STATIC | DYNAMIC
NO REFRESH
REFRESH

A mystery object exercise



remember about the item. It could be a stick of RAM, an old network switch, two different types of wires, input/output devices, or an old floppy disk! Students write down what they can recall and compare that to mind maps, knowledge organisers,or notes.

Workload: Contact your IT network manager for old computers or spare parts.

Fill in the blanks

How: Another quick and easy retrieval task is to take a knowledge organiser or mind map and cover keywords. Students then try to recall what is missing. It is a slight cheat as the students have visual prompts present so perhaps this is more of a guided retrieval practice. I tend to use this as a follow-on from a previous lesson's retrieval practice task where gaps in knowledge and understanding were identified.

Workload: Laminate knowledge organiser and mind maps in advance to make them reusable, then use stickies or page flags to cover the words.

Talk to the duck!

How: Many Computer Science classrooms have adopted debugging code with the aid of the faithful rubber duck. Make the duck part of an easy retrieval task by asking students to talk through what they learnt last lesson, week or term with the duck. By bringing that information to mind, it is changing the way that information is stored and it's easier for students to recall later.

Workload: Use a screwdriver to remove the squeak and save teacher headaches!

Emoji links

How: Students link emojis to sections from a theory topic – this should take no more than five minutes to complete. Emojis should be purposefully selected, with a few random emojis to see what fun ideas the students can come up with. This retrieval task is also a nice example of incorporating dual coding in the classroom.

Extend: This task could be extended as homework, with students using a different-coloured pen to find further links using their knowledge organisers, mind maps, notes, or flashcards.

Workload: Reduce the teacher workload by setting the challenge as homework, to design the emoji grids based on a given topic.

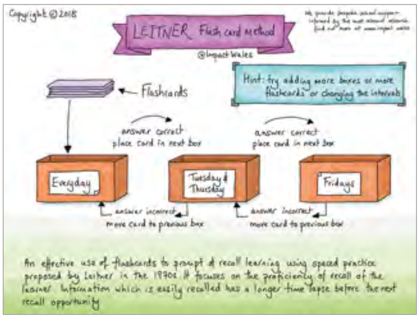
Flash cards

How: Following the Leitner flash card method using spaced practice and recalling

TOP TIPS

- Make tasks universal, so the activity can be easily modified across topics.
- Simplicity is key - quick to complete so it does not dominate the lesson.
- Tasks should involve everyone - each activity can be completed individually, in pairs or small groups.
- Duration - each task varies between 3-10 minutes. Students may well take longer at first, but once the task has been used a couple of times in lessons, the pace should quicken.
- Teachers - tasks need to be low-stake and should not require recording of results. It's important to circulate tasks to observe any common misconceptions.
- Feedback - this is, of course, essential; without feedback, students don't know what they got correct or not! This does not mean more work for the teacher, though - the students should self-mark by comparing their answers to mark schemes, knowledge organisers (which are single A4 documents that as a rule that contain key basic facts and knowledge on a given topic), mind maps, flash cards or notes.

GEMMA MOINE
Passionate about Computer Science, Gemma is a Secondary Computer Science teacher at The British School Al Khubairat, in sunny Abu Dhabi, United Arab Emirates.
@BSAKComputing, **@BSAKAbuDhabi**


Copyright © 2018
LEITNER Flash card method
@ImpactWales
Flashcards
Hint: try adding more boxes or more flashcards or changing the intervals
answer correct place card in next box
answer correct place card in next box
Everyday
Tuesday & Thursday
Fridays
answer incorrect move card to previous box
answer incorrect move card to previous box
An effective use of flashcards to prompt & recall learning using spaced practice proposed by Leitner in the 1970s. It focuses on the proficiency of recall of the learner. Information which is easily recalled has a longer time lapse before the next recall opportunity

© Image courtesy of @ImpactWales

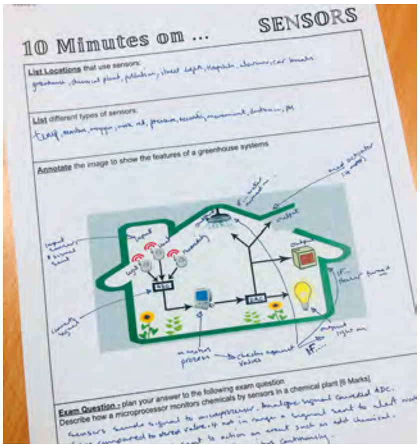
by writing answers down before turning flash cards over to check answers, is an effective method for retrieval practice. Impact Wales has an amazing poster to help guide students through this process, but students can use envelopes to store different piles. This task should take no more than 3-5 minutes of lesson time and is an excellent method to help prepare students for revision.

Workload: Students should create their own flash cards. The process of writing the flash cards helps reinforce learning.

Great packet race

How: Students race across the map, claiming states as they answer questions correctly. The map is generic and can be played on any topic. If played on a teacher's board, students can work in teams to answer questions. The game can also be printed and students can play in small groups. Questions and answers can be prepared, or students can use their knowledge organisers, mind maps, or flash cards to generate questions.

Workload: Use retrieval grids mentioned later in the article for the teacher question bank to reduce workload.


10 Minutes on ...
SENSORS
List Locations that use sensors:
List different types of sensors:
Annotate the image to show the features of a greenhouse systems
Exam Question: plan your answer to the following exam question
Describe how a microprocessor monitors chemicals by sensors in a chemical plant (6 Marks)

One of the '10 minutes on' sheets

Extend: Set as homework the task to annotate the map after the game, linking the name of tech states to theory topics and keywords.

Original idea: Magpied and adapted from **@SPBeale**, adapted by **@BSAKComputing**.

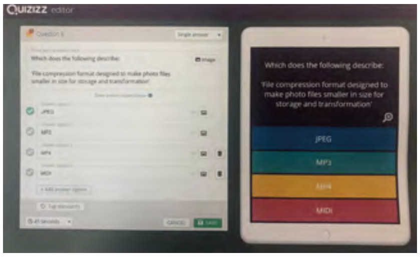
10 minutes on...

How: Students have ten minutes to list, explain, or compare one specific topic. Students also need to annotate an image which nicely incorporates dual coding, and plan how they would answer an exam-style question.

Extend: Students can complete as homework their final exam-style question/answer.

Quizizz - self-marking quiz

How: Create a free teacher account on **quizizz.com** and make a bank of quizzes on each topic needed. Questions are multiple


Quizizz editor
Which does the following describe:
File compression format designed to make photo files smaller in size for storage and transformation
JPEG
MP3
MP4
MIDI

Quizizz is a useful tool that allows for self-marking, whilst also letting teachers track progress

choice – allow for pictures and up to five possible answers. The quiz can be set as a live game, homework, or solo game, and can be played by students using a pin code or shared via Google Classroom or Remind. Quizzes should take 3-5 minutes for students to complete and the system self-marks. Quizizz also has a new feature called 'Classes' that allows for teachers to track students' progress.

Workload: You can search for existing quizzes on Quizizz to save and edit if needed. Once quizzes are saved they can be reused with very little preparation needed.

Retrieval grids

How: This task involves a little more planning, but retrieval grids are a flexible tool for low-stakes retrieval and feedback. Each box is colour-coded to represent whether the knowledge was learnt last lesson, last week, last month or way back! Simply add questions on one slide and answers on the following slide. Display the grid on the teacher's board or students' screens, then students individually fill in their answers on a printed blank grid or directly into their books. The answer grid can be displayed and students self-mark their work.

Workload: It can be time-consuming preparing this retrieval task, but you could set up a template at the beginning of the academic year with 20 blank question-and-answer slides. As you work through topics, you can add questions to different slides.

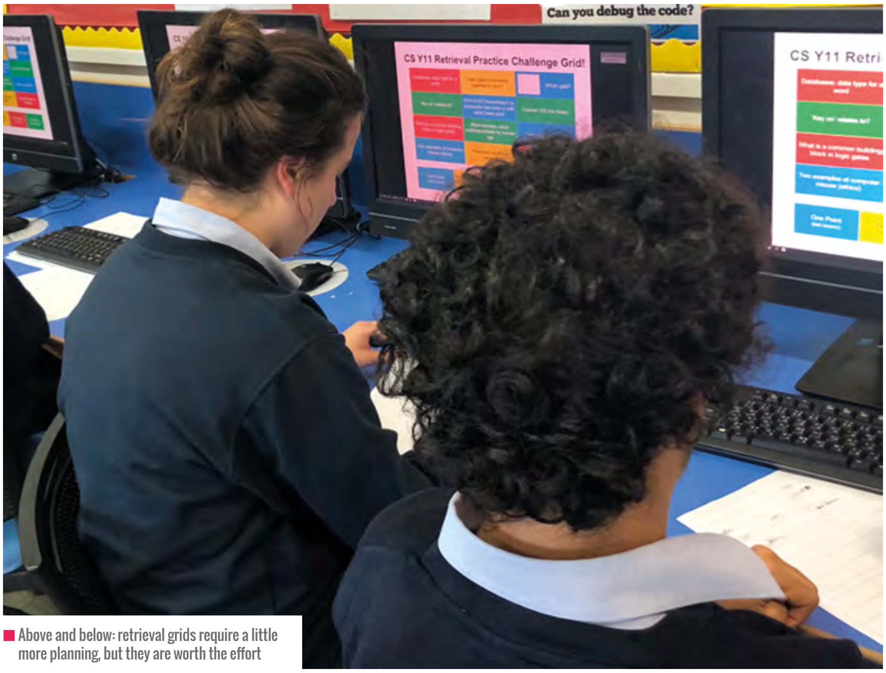
Extend: Ownership can also be taken from students themselves. My sixth form students were each given a blank template at the beginning of the year and asked periodically to update their own versions. You can also print and laminate the questions and answer grids back to back, and use periodically with individuals.

Original idea: Magpied and adapted from **@87History**.

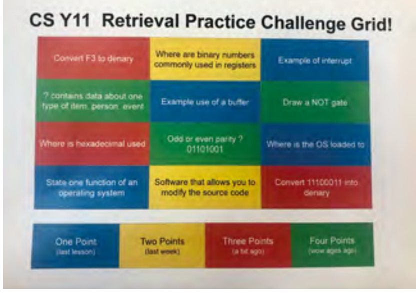
> IT'S BEEN REVEALED THAT THE MECHANICS OF MEMORY HAS A LARGE IMPACT ON LEARNING

THE GREAT PACKET RACE

THE GREAT PACKET RACE
B
A
C
B
C
A



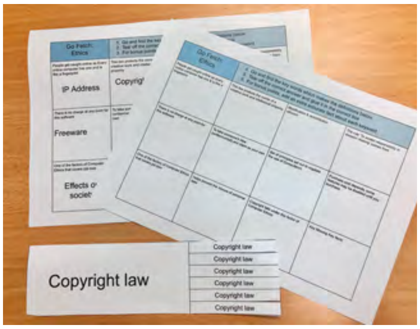
Above and below: retrieval grids require a little more planning, but they are worth the effort



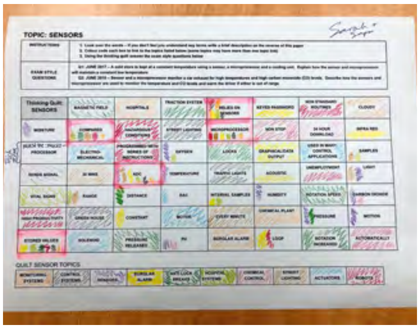
CS Y11 Retrieval Practice Challenge Grid!

Convert F3 to denary	Where are binary numbers commonly used in registers	Example of interrupt
? contains data about one type of item, person, event	Example use of a buffer	Draw a NOT gate
Where is hexadecimal used	Odd or even parity ? 01101001	Where is the OS loaded to
State one function of an operating system	Software that allows you to modify the source code	Convert 11100011 into denary

One Point (last lesson)	Two Points (last week)	Three Points (a bit ago)	Four Points (wow ages ago)



The 'Go Fetch' activity involves placing answers around the classroom, and gets people on their feet!



An example of a thinking quilt

Go fetch!

How: This retrieval task is a high-energy one that gets students moving around the classroom. Complete the template squares with a description or a key fact on part of a topic. Students race to find the corresponding answers on snippable sheets placed around the classroom. The answers are glued or written into the corresponding box, and students can add extra details for bonus points.

Extend: Make it a little harder and do not include all the answers on the snippable sheets.

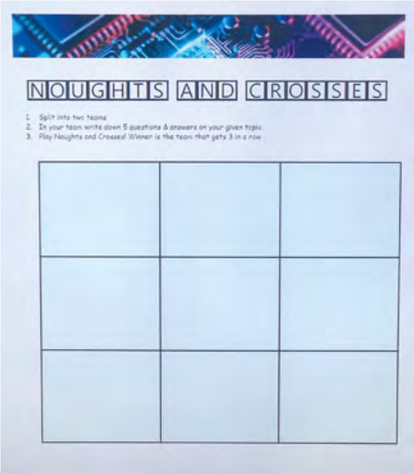
Original idea: Magpied and adapted from @SPBeale.

Thinking quilts

How: Students use the quilt topics along the bottom to identify related keywords on the main grid. Links between the grid and the topics should be the same colour, with some key words potentially sharing several topics. Students then use the grid to answer the exam-style questions. The task does have visual prompts and is not a complete recall task, but is an excellent revision resource.

Extend: Blank spots can be left for students to fill in additional keywords from their own recall, or a blank topic can be left along the bottom for students to identify.

Original idea: Magpied and adapted from @KKNTeachLearn.



NOUGHTS AND CROSSES

1. Split into two teams
2. In your team write down 5 questions & answers on your given topic
3. Play Noughts and Crosses! Winner is the team that gets 3 in a row

Noughts and crosses

How: Display a simple noughts and crosses grid on the teacher's interactive board. Split the class into two teams and get each one to write down five questions. Each team takes turns to answer questions, and if correct, places the team's allocated symbol, zero, or cross in a chosen space on the grid. The winning team is the one with three symbols in a row. Alternatively, students work in pairs with their own printed grid and play between two.

Speed it up: Previous homework could be for students to bring three questions and answers to the lesson, then the team selects five questions to use, or students could use their flash cards as question banks.

Original idea: Magpied and adapted from @BsaktL. (HW)

FURTHER READING

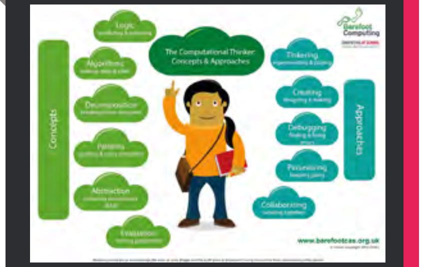
Retrieval practice is such a powerful technique - if you want to learn more about the topic, I would highly recommend the following books: *Make It Stick* by Peter Brown, Henry Roediger and Mark McDaniel, and *Powerful Teaching* by Pooja K. Agarwal and Patrice M. Bain.

Finding time to read can be difficult, so I would also recommend searching online for chapter summaries on the books. The **retrievalpractice.org** website is brimming with techniques and further information on the topic.

COMPUTING IN THE EARLY YEARS

Amanda Hubball and **Amy Stancer** report on the importance of an early start to children's computing education. They identify the challenges, affordances and good practice associated with teaching computing in the early years.

ADDED NOTES



Barefoot Computing – engaging, exciting and easy-to-teach resources to inject computational thinking into every corner of the curriculum with ease **https://www.barefootcomputing.org**

CAS Assessment is open to all CAS members. Through focus group activity it discusses current and potential good practice in the assessment of computing and reports back to the membership through the community forum, Schools Curriculum and Assessment Committee, CAS board and Hello World.

Computing in the early years is an enrichment which transcends the need for its own curriculum timetable. Computational thinking, which is in essence the key to young children's thinking and understanding in this area, forms a connection that joins curriculum content and provides skills for life. Along with all well-founded learning, practical engagement with the media is only the tip of the iceberg. The understanding, competence with language and skills will as much - if not more so, during the early years - be practised 'unplugged' as with a device. Recognising that much of the crucial learning, with a skilled teacher, can be undertaken in daily routines with existing resources, without heavy investment in expensive devices, is crucial to the embedding of effective teaching.

For example, when a child is taught the art of decomposition, they are learning the skills to solve practical problems which will affect their lives now and in the future. Whether looking at how to bake a cake or how to tackle global warming, a 'step at a time' approach, rather than the overwhelm of an unmanageable task, is the systematic and self-regulatory method which can lead to success. Computational thinking binds 'real life' with the school curriculum and provides children with a window on the world which can empower, and inspire.

Engagement

Positive engagement with computing in the early years is all immersive, it is led by the Characteristics of Effective Learning and the 'barefoot computing poster' is as relevant for under six year-olds as it is for Primary and Secondary. It is important that all educational transitions are considered and Early Years is a key stakeholder.

For Computing as a subject to be given credibility against maths and literacy, educators need educating. The training for Early Years Educators in computing has been sporadic to non-existent over the last 5 years. This creates a real professional knowledge gap for our teachers, who work within preschool and school education settings. This situation puts our early years teachers at a professional disadvantage and thus disempowers our young learners. Technology and the opportunities for learning don't stand still, yet investment in ensuring provision for children has. We believe that the recognition of the importance of computing should be evident in the confidence and ability of the teachers.

Children are not just inspired by subjects, but by relationships, connections and wonderings. If educators are delivering a vibrant and cutting edge computing curriculum, merged with cross curricular content that appeals to all children (regardless of passions for maths, arts, English, science and such like), an inspirational delivery and aspirational expectation, can hook children into computing.

> SECONDARY SCHOOL PUPILS WHO ARE NOT CHOOSING COMPUTER SCIENCE HAVE BEEN ON AN INVISIBLE JOURNEY

Gender issues exist because of preconceived ideas and societal influence that begins in the early years. If girls receive computational thinking skills through curriculum areas that interest them, rather than through an expectation that they will 'choose' this path in isolation, their passions can become fused with computing and misconceptions minimised. Secondary school pupils who are not choosing computer science, have been on an invisible journey, which has led them to a place of disengagement.

Experiences & Opportunities

Early Years provides the experiences and opportunities which relate to programming. It is these learning opportunities which formulate the basic pathway through the map of computing. Without these rich experiences with knowledgeable adults, the fluid transition is lost. We are part of the algorithm of programming learning and need to be seen as integral to later success.

The Early Years Amber Valley CAS Community, facilitated by Alfreton Nursery School and supported by Yvonne Walker (CAS Community Outreach Manager), are completing a review of a technology-enriched early years curriculum. The aim is to support the understanding and observation of computing-based learning within the Early Years.

Emanating from the key elements of the 'Barefoot Computing Poster' we intend to create accessible support for Early Years Teachers, creating reflective tools linked to Early Years Learning and offer opportunities to support current curriculum practices with a focus on recognising programming and computing competencies in young children. Part of these reflective tools will include an easy to use, quick reference monitoring and assessment system that can be incorporated into daily classroom life, as well as resources to support adult-child interactions. The work this CAS Community has undertaken is being supported by the 'CAS Assessment Working Group', led by John Woollard.

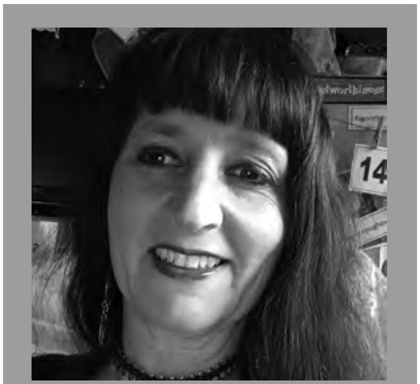
There is much work to do from the perspectives of national policy, initial teacher training, classroom delivery and pupil uptake to mention but a few, but the early years is the foundation on which to build. For the national picture of computing success in schools to truly change, early years needs to establish itself as the bedrock to high quality provision. For this to happen, national policy makers need to look again at the contribution of early years to computing education and give credence to its role in the journey of all future scientists, engineers, computer programmers…

Passionate Early Years practitioners who are driving this work forward, through the CAS Community include, Amanda Hubball, Amy Stancer, Ruth Davison, Joy Marson, Kerry Scotney, Amy Coleman and Ellie Smith. All highly skilled educators in the East Midlands, they hope to affect change for our youngest learners, enabling them to experience an inclusive and inspirational computing curriculum. (HW)

KEY LINKS:

Characteristics of Effective Learning
www.early-education.org.uk
supported by the DfE

Barefoot computing poster
https://community.computingatschool.org.uk/files/8221/original.pdf



AMANDA HUBBALL

is the computing subject lead at Alfreton Nursery School in Derbyshire. She is a teacher and Specialist Leader of Education for Alfreton Nursery Teaching School. She leads a Computing at School Community Hub and during the summer of 2019, Amanda facilitated an Assessment Working Group meeting, welcoming passionate early years professionals, all dedicated to developing computational thinking with young children.

AMY STANCER

is the Head Teacher at St Giles Nursery School in Lincolnshire. She has been passionate about the teaching of computing for many years and has led her school to an excellence in provision. Amy has had her work on computing published previously and is committed to the integration of high quality computational thinking teaching across the early years.

AN ARTIST IN EVERY PROGRAMMER

Why creative coding is a great start for teaching Computer Science

In a recent programming workshop, one of my pupils had some experience with JavaScript. He didn't want to program with blocks, because it felt to him like a step back. He felt that he had surpassed this 'child language'. It was challenging for us to motivate him to participate. Later in the workshop, we introduced our Processing editor and started to program animations. He was instantly thrilled about the animations and suddenly he didn't mind using the 'childish' block programming.

We often encounter this perception of Scratch-like block programming as a toy for younger children. Especially so from older kids, and those who have some experience with text languages. They often prefer to use 'real' programming languages, the ones that are used by adults and developers. Arguments about the didactic virtues of Scratch don't work against that perception.

Building new programming tools

Based on these experiences, we had the idea of developing tools and environments that combine the advantages of block programming with 'real' applications or languages. Applications that are popular with children (such as games), or something they'd want to create themselves.

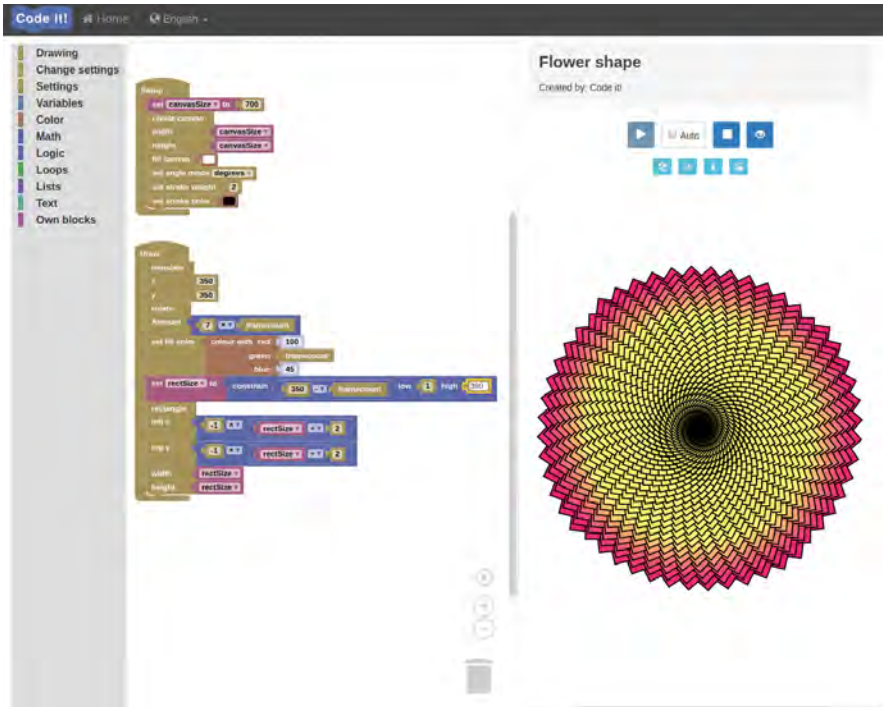
We wanted to build programming tools that are fun, easy, and quick to learn. Tools that allow plentiful creative expression, with results that delight and motivate children. The first programming environment we built was our Duck Race makerspace, where kids can create their own games. Most children love to develop games, but the creative possibilities of this makerspace are limited.

In the search for other applications that have great creative potential, we discovered Processing, which is a framework specifically designed for artists. The idea behind it is to provide non-programmers a tool with which they can easily create images and animations of high visual and aesthetic quality. Processing is, in its own way, a new field of programming: creative coding, and the artistic expression through programming.

Creative coding made easy

We built an editor in which Processing can be programmed with blocks. We use block programming because it makes it easy to get started with coding. The editor allows you to create impressive pictures with just a few blocks. With every new block and every new programming concept, the creative possibilities multiply. The children in our workshops, girls and boys alike, were enthusiastic about their pictures and animations. We believe that this affinity to pictures is due to them being able to share their images via social media services such as Instagram.

A Processing program is about drawing shapes on a canvas, producing either an image or an animation. The basic structure of a Processing program is made up of a Start and a Draw function. The Start function



The block-based editor for the Processing framework enables impressive images and animations

ANDREAS KOCH

Andreas Koch teaches CS in schools and after-school workshops, and is one of the founders of Code it!

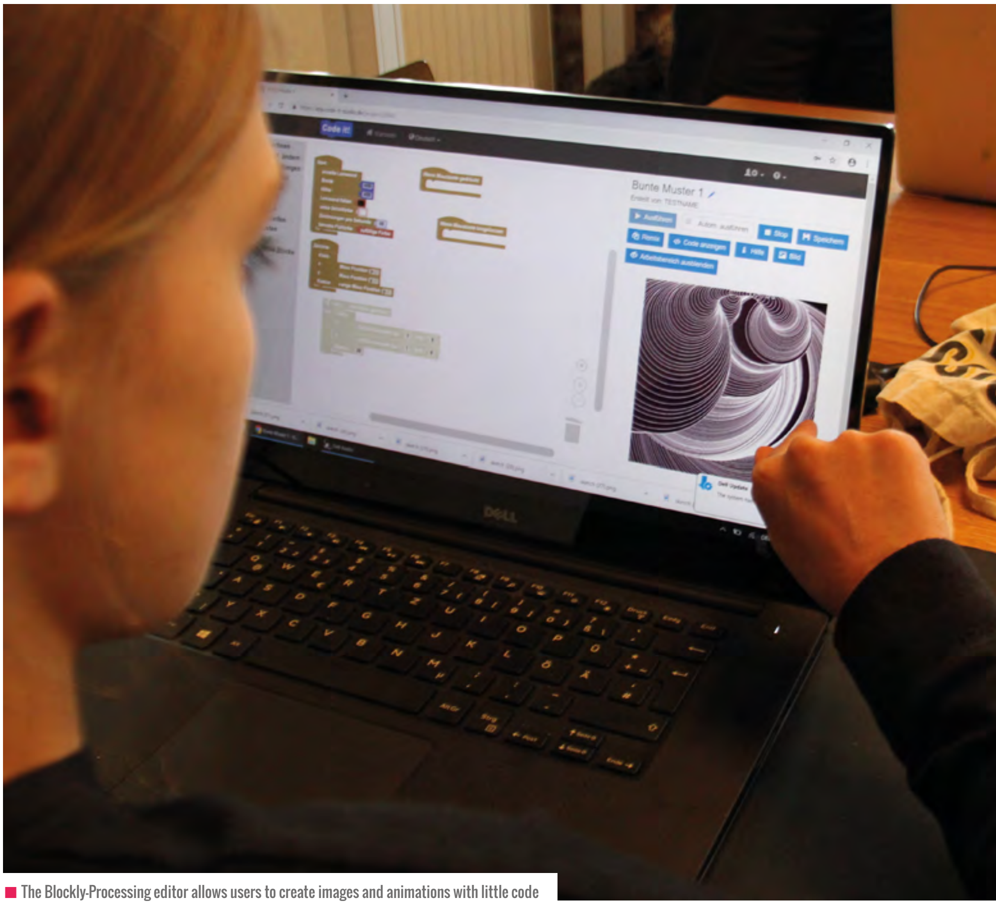
contains all features that remain static throughout the program – the size of the canvas, the number of drawings per second, the global variables, and suchlike. The Draw function contains the shapes to be drawn, their properties (colour and size, for instance), and instructions on how to animate them. A circle is drawn with only one block. Just a few blocks more and the circle is animated or follows the position of the mouse.

Our editor doesn't provide all Processing functions. Following the didactic reduction method, we limit the editor to the essential drawing functions (e.g. forms, loops, logic). In the future we will provide further functions that can be loaded as extensions. Functions for photos, videos, and sounds, along with machine learning, are planned. The editor also allows an easy transition to text programming. The block program can be displayed as source code and can be exported to the Processing web editor.

In August we published an Hour of Code to introduce creative coding and our Processing editor, and teacher materials. We're always happy to have suggestions and ideas on how to improve the tool.

Get inspired

Besides being well suited for programming education, Processing is also suitable for projects in many other areas/subjects. It can be used for art classes to teach about space, colour, shape, perspective,



The Blockly-Processing editor allows users to create images and animations with little code

proportion, or aesthetics; in maths classes you can use it for algebra and geometry; in physics classes it can provide simulations and illustrations. There are countless projects on the internet that can serve as templates: Instagram filters, Piet Mondrian generators, picture stories, video animations, art installations, or even machine-learning applications.

> A LARGE COMMUNITY OF ARTISTS, ACTIVISTS, AND EDUCATORS HAS EMERGED AROUND PROCESSING

A large community of artists, activists, and educators has emerged around Processing. You can find many pictures, animations, art projects, and more on the web – together a great source of inspiration for new projects. The children can become part of that community by publishing their finished their pictures and animations on a website, Instagram, or other platforms. This gives the children feedback and recognition for their work, which in turn motivates them.

The Processing Foundation actively promotes the use of Processing for educational purposes – Saber Khan is the Education Community Director. Teachers can find a variety of teaching materials and inspirations online. For example, the excellent introductory books and tutorials from Daniel Shiffman, who also runs one of the most entertaining programming YouTube channels on the web (**youtube.com/user/shiffman**), or the teaching material by CSforALL. (HW)

Pinus nigra trees at Mt Barker, Canterbury, New Zealand. Some of the larger trees have cones ready to release seeds for dispersal

EXPLORING THE INTERFACE OF ECOLOGY, MATHEMATICS, AND DIGITAL MAKING

Ecologists have the most fun – what other subject lets you combine tree spotting (and climbing) and programming to help understand and protect our natural world?

Seed dispersal, or the movement of seeds from a maternal plant to a new location where a child plant can germinate, take root and grow, is important because it allows plant populations to reach new locations and habitats. Seeds that are well adapted for wind dispersal have structures that increase drag, making them fall slowly so that they spend more time in the air. This means that they are more likely to travel further, increasing their dispersal distance. Being able to predict dispersal distances is fundamental to understanding how plant populations can spread across landscapes, whether you are trying to control a pest species, or trying to conserve a plant in the face of rapid climate change.

From ecology to code

Seed terminal velocity is a key biological trait. To explain, it influences a plant's potential dispersal distance, alongside the height from which the seed is released, and the strength, direction, and turbulence of the wind blowing at the time of release.

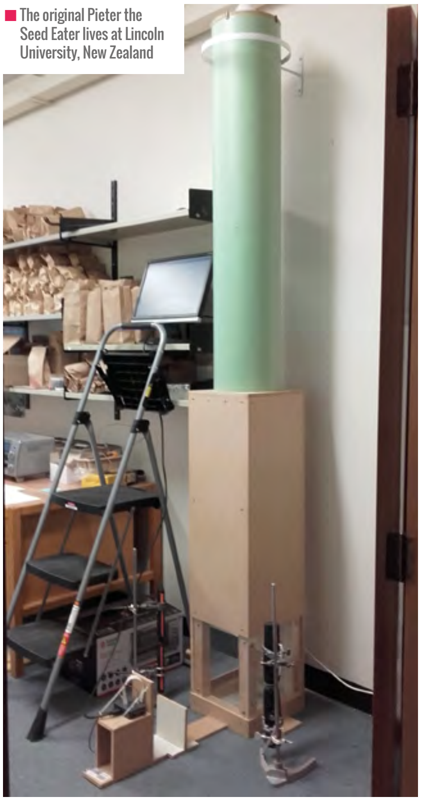
But how do we measure it? There are obvious problems with using a stopwatch to record falling times. And what if air currents mean the seed doesn't fall in a straight line? We use a device called 'Pieter the Seed Eater' (pictured). Pieter uses a tall pipe fitted with a Raspberry Pi and Camera Module to capture and process images of a falling seed, to accurately calculate the seed's terminal velocity.

Pieter the Seed Eater was designed to measure the terminal velocity of pine (Pinus species) seeds from invasive trees in New Zealand, allowing us to assess the effect of slow and fast seeds on dispersal distances, and predict invasive species spread.

Support and resources

The biosciences fall significantly behind other STEM subjects in employability statistics,

Pinus nigra cone at Mt Barker, Canterbury, New Zealand. This cone has released its seeds already



The original Pieter the Seed Eater lives at Lincoln University, New Zealand

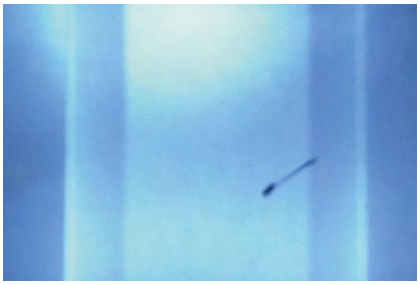
© Images taken by Sarah Wyse and Pieter the Seed Eater

> PIETER THE SEED EATER WAS DESIGNED TO MEASURE THE TERMINAL VELOCITY OF PINE SEEDS

yet the study and practice of biology is increasingly quantitative and computational. We want to highlight the interdisciplinary nature of biology by providing opportunities to wear some of the many different hats a bioscientist may wear: from traditional scientist to natural historian, coder, statistician, mathematical modeller, creative thinker, and engineer.

Resources to support teaching activities across this spectrum will be available this autumn, taking students from tree identification and seed collection, through the creation of their own Seed Eater to measure terminal velocity, to considering the implications of seed physiology and velocity on the potential spread of plant species across landscapes. Students will also be able to contribute their data to an online database and explore how their data fits into the global dataset. Keep an eye on the November issue of *Hello World* for more details.

You can follow the whole series from start to finish, or focus on relevant parts of the project: for example, ecology and biology, probability and variation, trigonometry and falling objects, or digital making to measure the terminal velocity of falling objects. The resources will allow groups to engage at a range of levels of understanding, with the first edition aimed at Key Stage 3 (age 11-14). Ultimately, we would like to reach Key Stage 1 to sixth form, and welcome feedback and engagement with the project from anyone who is interested in taking part. (HW)



A Pinus pinaster seed - time-lapse images are used to calculate the speed at which the seed is falling

PEN HOLLAND AND SARAH WYSE

Dr Pen Holland (@Anaspene) is a Lecturer in Ecology, and RPi Certified Educator, at the University of York, UK. Pen is a quantitative ecologist whose interests range from using simple models to understand complex ecological problems, to developing novel ways to teach, disseminate research, and engage people in the biosciences.

Dr Sarah Wyse (@SarahtheWyse) is a postdoctoral research fellow at the Bio-Protection Research Centre, based at Lincoln University, NZ. Sarah is a plant ecologist whose current research focus is on quantifying seed dispersal potential to model invasion risk of introduced conifer species in New Zealand.

FIVE YEARS ON, REVISITING CRAZY CHARACTERS

Reviewing a lesson activity using semantic waves

Crazy Characters is a free online lesson plan which introduces algorithms to primary pupils using an unplugged activity. It is one of the free resources available from the Barefoot website [1]. In the activity, learners are asked to follow verbal instructions (see Figure 1) to draw a crazy, made-up, character. The instructions are not very precise so that learners can then improve the algorithm. Following a whole-class activity, learners then design their own algorithm in order to draw their own crazy character.

JANE WAITE
Jane Waite works for Queen Mary University of London. She undertakes research in computer science education, and provides teacher training and support for resource development. Jane worked in the IT industry for twenty years, and was a primary school teacher for ten years.

KARL MATON
Karl Maton is Director of the 'LCT Centre for Knowledge-Building' at the University of Sydney. Karl is the creator of Legitimation Code Theory (LCT), which is being widely used to shape research and practice in education, sociology, and linguistics.

LUCINDA TUTTIETT
Lucinda Tuttiett is the Barefoot Education and Liaison Manager with the South West Grid for Learning. She taught in primary schools for 18 years, before moving into the Advisory services in Bristol and Somerset supporting schools with ICT, and then the Computing curriculum.

Where did Crazy Characters come from?

In 2012, Michael Gove disapplied the English ICT Curriculum, creating a two-year hiatus when we, primary teachers, had to await a new statutory ICT Curriculum.

In the meantime, we were still required to deliver the old curriculum or to start to teach what we thought might come next. I recall being at BETT, the big computing education trade show, as the announcement was made and then frantically searching for resources and people who could help me rewrite my school's ICT subject planning. I recently found my first revised scheme of work, which I created for September 2012 – there was no mention of algorithms, but there were learning objectives such as, 'I can predict the results of someone else's instructions'.

Many computing lesson plan versions later, in Spring 2014, I applied for a secondment from my school and for a job as a content author on the Barefoot Computing Programme. Managed by the British Computer Society (BCS), the initiative was funded by the DfE and BT, and was one of the first of many successful, innovative, and crucial Computing At School (CAS) programmes to support teachers in their delivery of computer science in school. Very luckily, I got the job and my life changed completely.

Over the next year, the Barefoot team developed resources that demystified the computer science elements of the new Computing Curriculum. Using an iterative approach, we wrote concept documents and their associated classroom activities, publishing as we went along. The algorithms concept was first, and I was tasked with thinking of an introductory unplugged activity. In June 2014, Crazy Characters was born.

Crazy Characters was one of the first resources on the new Barefoot website, part of the very first continuing professional development (CPD) presentation, and is still a staple of the Barefoot volunteer workshop delivered to teachers in schools. When writing the activity, I was keen to make sure

GRAVITY EXAMPLES

An activity with weaker semantic gravity would be to ask learners to memorise a definition of an algorithm without any context, such as `an algorithm is a set of precise rules or steps to solve a problem'. Semantic gravity would become stronger by adding an example, such as 'an algorithm is a set of precise rules or steps to solve a problem such as an unambiguous set of steps to draw a square'. This activity shifted from weaker to stronger semantic gravity. It would be strengthened further if learners then engaged in a practical activity of creating algorithms to draw squares where the need for equal length sides was explored to highlight the importance of precision.

that it was easy to run in class, fun and, most importantly, gently introduced this new word ‘algorithm’ by doing, rather than telling.

I built on the way in which I normally taught instruction writing in literacy, which included a spot of curiosity, teachers getting things wrong, humour, and peer review. I had no idea that five years later I would be asking Professor Karl Maton, a leading education researcher, to review the lesson plan and to investigate my planning in terms of what is called semantic waves.

Semantic waves, what is that?

I was introduced to semantic waves by Professor Paul Curzon, who has written about their potential for teaching programming [2]. The notion of semantic waves is part of a wider theory called Legitimation Code Theory or ‘LCT’ (not to be confused at all with the coding of programming), created and developed by Karl Maton[3]. Very simply put, we can use semantic waves to review learning activities, and abstract the process of learning to better think about how learners develop an understanding of knowledge [4]. The overall aim is that, by doing this, we can reflect on and improve teaching experiences for our students. To explain these ideas, I need to briefly introduce two concepts: semantic gravity and semantic density.

> SEMANTIC GRAVITY EXPLORES THE CONTEXT OF MEANINGS AND HOW MUCH MEANING DEPENDS ON THE SOCIAL CONTEXT TO MAKE SENSE

Semantic gravity explores the context of meanings and how much meaning depends on the social context to make sense. So where meanings are more concrete (e.g. practical examples or personal experience) semantic gravity is stronger and where meanings are more abstract (such as theory), semantic gravity is weaker. Changes in semantic gravity can be shown over time, such as when teachers or students move from theory to examples or from practical activities to a concept.

DENSITY

An activity asking learners to `to follow the instructions to draw a square’ would have a weaker semantic density than one requiring learners `to follow the algorithm to draw a square’. This is because the first activity is less complex, as the term instruction has a less complex meaning than the term algorithm.

Semantic density explores the complexity of meanings. Where meanings are relatively simple, such as describing something in everyday language, semantic density is weaker and where meanings are more complex, such as using technical concepts, semantic density is stronger.

We can depict changes in semantic gravity and semantic density as a semantic profile, an example is shown in **Figure 1**.

In this example from teaching Biology, the teacher begins by discussing a scientific concept in abstract and technical terms. The teacher and students then unpack some of its meanings in everyday language through practical and concrete examples. Finally, the students repack those examples into technical terms by completing a table of concepts. This moves from abstract and complex meanings down to more grounded and simpler meanings, and then back up to abstract and complex meanings. This kind of movements up and down are called ‘semantic waves’, and a rapidly growing body of research is showing that they are crucial for knowledge building in classrooms. Study after study is showing that waves enable knowledge to be built, while flatlines (such as continuous description or incessant theorising) hinder knowledge building.

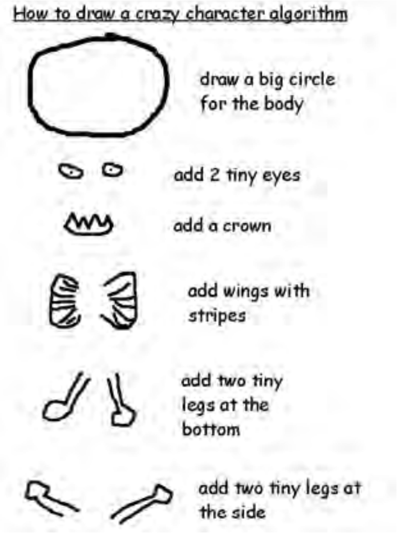

How to draw a crazy character algorithm
draw a big circle for the body
add 2 tiny eyes
add a crown
add wings with stripes
add two tiny legs at the bottom
add two tiny legs at the side

Figure 1 - Teachers read out their algorithm of how to draw a Crazy Character *Image: Jane Waite*

These insights are now feeding into teacher training, curriculum planning, and classroom practice.

Enough of the theory, we now need a concrete example, so we are going to strengthen our own semantic gravity!

How did we create the semantic profile for Crazy Characters?

The creator of ‘semantic waves’ is Karl Maton, Professor and Director of the LCT Centre for Knowledge-Building at the University of Sydney, in Australia [5]. I contacted Karl and asked him if he could help me ‘semantic wave’ a resource, and suggested Crazy Characters as it is very familiar to me, is still very popular with teachers, and is also due for a review.

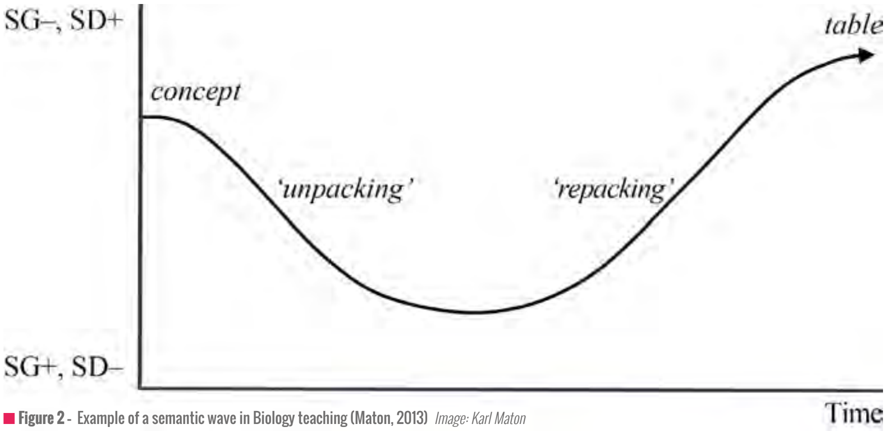
In an online hangout, Karl read the lesson plan, and together we walked very carefully through each step of it and drew up the semantic profile. We have profiled the plan, as though a teacher was following the plan to the letter.

A USEFUL LINK

To get the most out of this article, download the Crazy Character lesson plan and walk through it with us **www.barefootcomputing.org/resources/crazy-character-algorithms**.

What does the semantic profile for Crazy Characters look like?

The semantic profile for just the introduction part of the Crazy Characters lesson is shown in **Figure 2**. It is broadly a U-shape, but with steps coming out of the U. We next go through each of the lesson plan steps and explain the wave.



■ **Figure 2** - Example of a semantic wave in Biology teaching (Maton, 2013) *Image: Karl Maton*

SIGNALLING To start with, the teacher is asked to explain to students that a special new word is going to be used. Learners are signalled that something important is coming, that a concept high up the semantic profile is on the way. Learners are NOT provided with a definition at this stage. Instead, curiosity and expectancy are kept high, so they can form their own understanding of the term later through the practical experience. There is no practical concrete activity going on here (so semantic gravity is weaker).

CONCEPT INTRODUCTION The term 'algorithm' is introduced as the teacher starts to use the word; the teacher should NOT explain what the word means at this point. There is no practical activity here (weaker semantic gravity), but it is clear that the term is a complex and technical one (stronger semantic density).

CONNECTING In the plan, the teacher is instructed to say they are going to use the algorithm now. This clear connection of the concept to the activity is very important. The connection enables learners to add the knowledge they gain during the practical activity to their emerging understanding of the meaning of the concept. As shown in **Figure 2**, the semantic profile line drops, like a bungee rope, as we connect the theory to the practical activity (strengthening the semantic gravity as the context is introduced). If there was no connection, the line on the profile would break.

ADVANTAGE

Some research indicates that learners from more socially advantaged homes may be more comfortable with semantic waves than students from less advantaged homes, who may experience less of semantic waving. The rationale being that some learners are more likely to have generalised and complex meanings explained to them, from a very young age. In other words, the `why' question gets answered, and experiences are provided that exemplify the` why'.

CONCRETE ACTIVITY Next, the teacher is asked to read out the steps to enable the learners to draw the crazy character. The wave is low on the profile: it is a concrete activity (stronger semantic gravity) and likely to be expressed through relatively simple meanings (weaker semantic density) ... unless learners start to use the term 'algorithm', in which case there would be little spikes of semantic density.

COUNTER EXPECTANCY The teacher is instructed to be very vague with the instructions given to learners. The aim is that when she asks the pupils to share their drawings, the image will be very different and she can say that she did not expect this to be the case and ask why. This is called counter expectancy. This means that the context in which the learners are developing their understanding is challenged and alternative options are raised. This increases the meaning of the concept. On the semantic profile, this is shown as a step up (widening the context weakens semantic gravity; adding meaning strengthens semantic density).

STAGED RETURN Next, the teacher is required to ask the learners how they could improve the algorithm. Learners start to think about making the algorithm more precise but this is still in a relatively specific context. On the graph, this shows as another step upwards (adding meaning strengthens semantic density).

PACKING Finally, the lesson plan instructs the teacher to ask a generic question of `What was the algorithm?' This is a more general view of the activity requiring the learner to 'pack' their accumulated understanding from the practical activity. Again, this is moving up the profile, further away from a specific context and adding more meanings (reducing context weakens semantic gravity).

THE REST OF THE LESSON We have not profiled the rest of the lesson for this article. Broadly, it follows a similar set of patterns. However, the highly prescriptive nature of the introduction is loosened as the learners create their own Crazy Character algorithms. Included in this is the introduction of a further concept, that of debugging, as they ask their friends to implement their algorithms as drawings and then, together, they debug the algorithm in order to produce the same imagined character.

How has creating the semantic profile been useful?

By drawing the semantic profile for Crazy Characters, with Karl Maton, I have had the opportunity to apply semantic waves, from theory to practise. This experience has provided a number of useful outcomes.

Firstly, it has introduced me to a language that has helped me describe the lesson plan. Secondly, and more importantly, semantic profiling enabled me to analyse and reveal why the learning activity worked, showing how ideas were introduced in a concrete way and more complex meanings were gradually added, stepwise, to develop a more general and abstract understanding. Thirdly, the process has supported my review of the activity, helping me think of ideas to improve and build upon the lesson plan. Finally, I have concluded that semantic profiling is a practical and useful approach which I will continue to explore and use when designing teacher professional development and in creating lesson planning material.

Would you like to change Crazy Characters?

To maintain the semantic wave, I would like to add a follow-on lesson which applies what was learned in this unplugged lesson in a programming context. I would also like to reorder the learning intentions to match the use, modify, create theory [6] and I would increase the use of the term design, following my own research area [7].

However, overall, I would not change the main steps of the Crazy Character lesson plan. By creating the semantic profile, I have revealed how the plan provides a carefully scaffolded learning experience to help learners develop an understanding of the meaning of the algorithm concept. As shown in **Figure 2**, the lesson plan includes a signal that a new concept is to be taught, introduces the concept, connects theory to a concrete activity, incorporates a concrete activity and increases the meanings condensed within the concept to reveal a 'packed' (complex) definition of the concept.

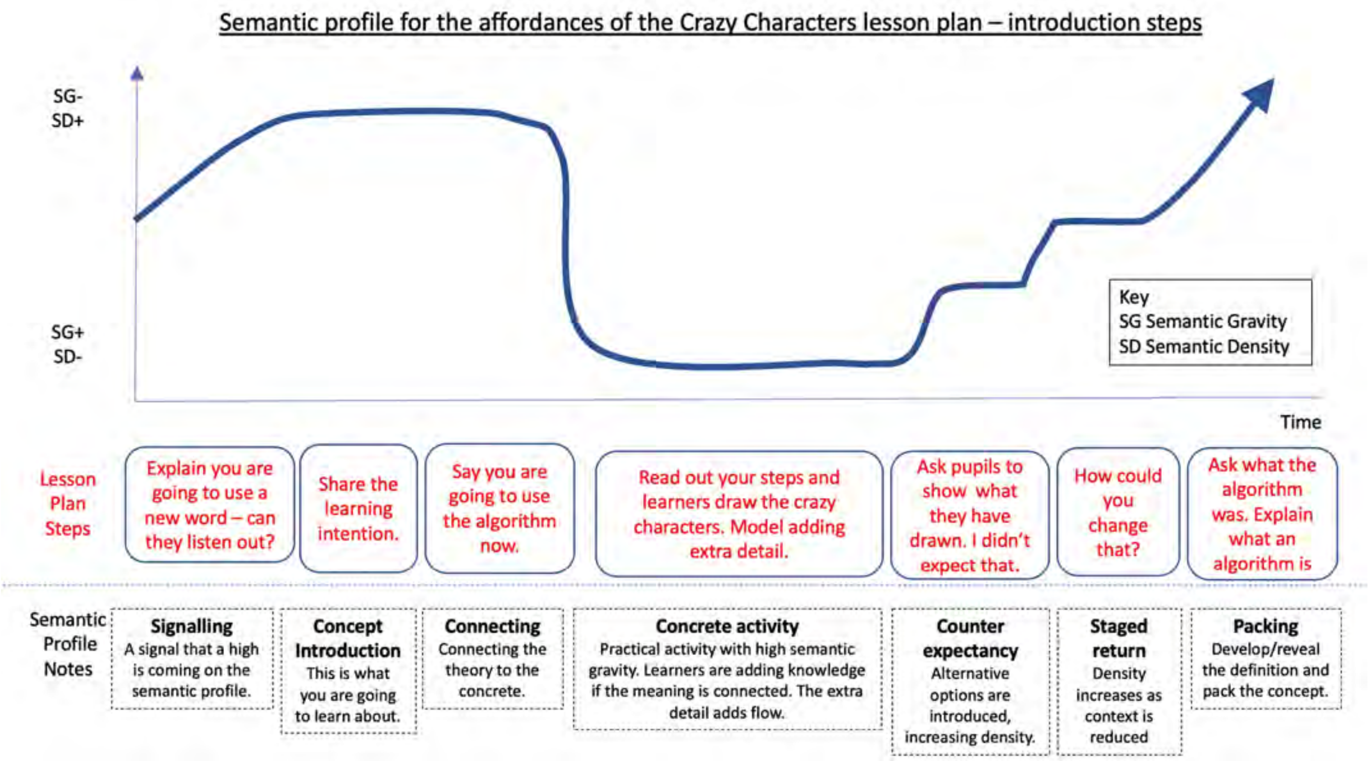
Working with Karl Maton on reviewing Crazy Characters has been thoroughly enjoyable. I am indebted for the time he has kindly spent supporting me in writing this article. I would also like to thank Lucinda Tuttiett who read through the article and helped keep it real!

PROFILES

Does all this mean that the profile for Crazy Characters will always be the same?

No. The profile is likely to be different each time it is delivered. We have analysed the lesson plan as though it is delivered to the letter of the plan. Teachers are likely to change how they deliver the lesson, therefore the semantic profile will be different each time they deliver it. Similarly, different learners will engage in an activity in different ways, this will mean that each learner experiences a different personal semantic profile based on their own knowledge building event.

I now have a fledgling understanding of how semantic waves can be used to reflect on and develop teaching activities, and we hope that by sharing our semantic profile of a popular lesson plan, we will help others learn about this approach. (HW)



Lesson Plan Steps	Explain you are going to use a new word – can they listen out?	Share the learning intention.	Say you are going to use the algorithm now.	Read out your steps and learners draw the crazy characters. Model adding extra detail.	Ask pupils to show what they have drawn. I didn't expect that.	How could you change that?	Ask what the algorithm was. Explain what an algorithm is
Semantic Profile Notes	**Signalling** A signal that a high is coming on the semantic profile.	**Concept Introduction** This is what you are going to learn about.	**Connecting** Connecting the theory to the concrete.	**Concrete activity** Practical activity with high semantic gravity. Learners are adding knowledge if the meaning is connected. The extra detail adds flow.	**Counter expectancy** Alternative options are introduced, increasing density.	**Staged return** Density increases as context is reduced	**Packing** Develop/reveal the definition and pack the concept.

Figure 3 - Semantic profile for the Crazy Characters lesson plan - introduction only *Image: Karl Maton & Jane Waite*

A hack:jam, open to students in years 7 to 10, that took place back in June 2018

INCREASING ENGAGEMENT IN COMPUTER SCIENCE

How a challenge-packed curriculum has led one school to boost its GCSE Computer Science take-up

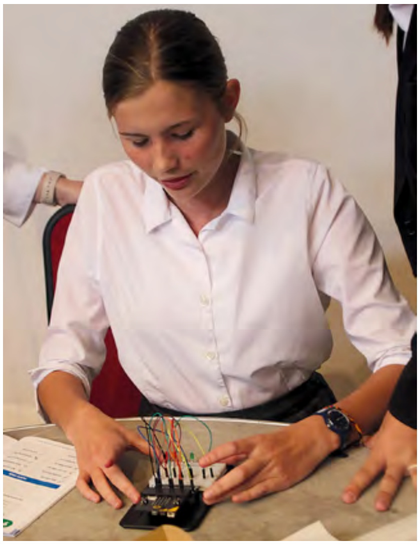
Over the past few years, there's been a marked change in the curriculum that we have offered throughout Key Stage 3. This has been driven, in part, by the move from ICT to Computer Science at both GCSE and A-level. In order to enable students to make an informed choice as to which GCSE options to take, we felt it important to give them as much exposure as possible to the types of challenges they will undertake as part of Key Stage 4 studies in Computer Science.

In Year 7 the main aim of our course is to encourage the development of computational thinking skills. We very much see this as a way to develop logical thinking skills, rather than students studying coding. There's a subtle difference between these two concepts. If students aren't naturally engaged in the use of computers, introducing coding straight away can cause some to feel disenfranchised. Computational thinking can be delivered away from coding initially. Logic puzzles and escape room-style activities all engage students, and still deliver the core learning objectives.

At the end of Year 7, the aim is for all students to be able to apply concepts of decomposition, abstraction, pattern identification, and algorithmic thinking to a range of problems. The end of year assessment has been designed to assess these skills. Rather than students being required to state theory, they instead apply their skills to a range of challenges. Students find this engaging, and some even enjoy the assessment experience!

In Year 8 we introduce students to text-based languages. We use Python as our preferred language (we like that it's quite an accessible language for students to learn, and that the colour-coding helps with error checking, amongst many other reasons), although students also gain exposure to mark-up languages in a web design unit which we deliver. Students also learn the importance of the binary number system, and why this relates to computer use. They produce videos explaining the theory and how to convert between different systems. More able students have also previously included binary addition in their videos. They really enjoy the filming, acting, and editing elements of this task. Many show great pride in their work, and we have showcased the best work through playing them as part of the end of school assembly.

In Year 9 students used to study for an entry level qualification in Computer Science. To some extent we found this qualification a little restrictive. We had to very much teach to the specification, rather than being able to give a broader, more differentiated



experience to students. With the deadline for submission being mid-May, students also completed the qualification several weeks before their Year 9 lessons ended.

iDEA badges

Last September we changed to the iDEA programme. This involved students completing a number of badges in order to achieve their Bronze award. We redesigned our Year 9 curriculum around this qualification, and have found that students have really enjoyed the more varied course this enabled us to offer. It has also had a major impact on the number of students choosing to study GCSE Computer Science. Next year, over 45% of the year group have chosen to study the subject at GCSE level, an increase of 20% over previous years. It's been pleasing that much of this increase has come from female students.

The iDEA badges have largely been set as homework activities which delivers students the base level understanding of key elements of theory, that they then put into practice during the lesson. For example, students initially programmed an e-safety quiz in Python. The e-safety theory was covered in several badges, and that was then incorporated into the quiz which they programmed in their lessons. They then created a website to promote the quiz and give people help and advice. The website badge helped them to understand the basics of HTML and CSS, prior to actually coding their website using Notepad during their lessons.

Small teams were given a problem to solve

For one half term per year group, students complete a micro:bit-based unit which gives them hands-on experience of directly manipulating hardware. This year we have introduced the use of the :MOVE mini, which is a micro:bit-controlled buggy. Students have built the buggy themselves, and have then produced programs to guide it around a circuit that's been laid out in the classroom using insulation tape. Year 9 students have used the Kitronik Inventors kit to produce a range of different circuits. This kit consists of a breadboard which they can wire up to a range of external components. Students really enjoy these hands-on style activities. The micro:bit offers really good potential for differentiation, as it can be coded using block code, JavaScript, or Python.

Competitive challenges

In June 2018 we launched our first ever hack:jam. This was run as a competitive event in the school theatre. It was run in an evening, and was open to Year 7 to 10 students. Students were able to choose their own teams of three or four. Each team had a range of pens, pencils, cardboard, micro:bit, and a laptop. The starting exercise was a game of consequences, where initial ideas for their product was generated. This led to some interesting potential concepts, including an alarm which alerted you that you had put your phone into a washing machine. Students then produced a solution to their problem. This was completed in two hours, and was very much a prototype rather than a fully functional solution. Each team then had three minutes to present their solution to their peers and judges. Teams were judged on merit rather than age, with the eventual winners being a Year 7 group.

With the event being so popular, we also ran a similar one for our 8- to 11-year-olds. All of the year group from the prep school took part. This meant that, when they started their Computer Science lessons the following September, they were already excited about studying the subject.

I believe that the way that we have redesigned our curriculum to be far more kinaesthetic and hands-on, rather than just delivering traditional coding lessons, has really enthused students and encouraged a much larger increase in GCSE numbers (a 42% increase last year in students opting to take CS, and 63% this year, with a very pleasing increase in the number of female students studying the subject). This has naturally also led to increased numbers at A-level. All students still learn to code in text-based languages but, through introducing them via the hands-on activities, all have been far more engaged. The iDEA programme has been a really positive move in Year 9 as everyone is still working towards a qualification, whilst also giving us far more flexibility in the design of our curriculum. (HW)

GARETH EDGELL

Gareth Edgell is Head of Computer Science at Kingswood School in Bath. After spending ten years in industry, he changed careers to enter the teaching profession. Gareth introduced Computer Science at GCSE level into Kingswood, and also oversaw the transition from ICT to Computer Science. Over the past five years Gareth has been involved with Computer Science assessment and qualification development, both in the UK and internationally.

WHY WE SHOULD TEACH CHILDREN TO CODE

Simon Peyton Jones explains the numerous benefits of teaching coding to children

In his March 2019 blog post, Andreas Schleicher, Director of Education at the OECD, asks "Should schools teach coding?", somewhat mis-reported in the press as "Teaching children coding is a waste of time, OECD chief says". But it's a good question. You can find the links to both articles on page 64.

Let's start at the beginning, though. Technology moves fast. To equip our young people to flourish in a world of change, we therefore strive to give them a foundational understanding of the world that surrounds them, and an intellectual toolbox that equips them to deal with successive waves of technology. For that reason, the new computing curriculum in England, introduced in September 2014, establishes computer science (not just coding, and with computational thinking at its core) as a foundational subject that all children learn, alongside maths and natural science, from primary school onwards. The previous Information and Communications Technology (ICT) curriculum was focused on, well, technology. The new curriculum is focused on ideas and principles. As the famous aphorism puts it, "computer science is no more about computers than astronomy is about telescopes".

But if computer science is not about computers, what is it about? It is the study of information, computation, and communication. Take information, for example: suppose I show you a picture of the French national flag, and one of the Mona Lisa, and ask 'which picture contains more information?'. What does that even mean? A way to make the question more precise might be, 'Suppose I dictated instructions to you over the phone; which picture would take longer for you to reproduce?'.

Clearly the Mona Lisa has more information in this sense: it would be a slow and painstaking job for me to dictate instructions to reproduce it at your end, even rather approximately. Thus, we have begun to speak of information as a measurable quantity. We start to think about how tightly we could compress data before transmitting it and how we could detect, and perhaps correct, errors made during transmission. All this is called information theory; it is part of computer science, it has a substantial body of theory, and it has immediate practical consequences.

Where coding fits in

What then, is the role of coding or programming (the terms are roughly equivalent) in computer science? Coding is not the message of the new computing curriculum: it is its medium. Coding is the labwork of computer science: it motivates, illuminates, and brings to life the dry bones of theory. Without programming, computer science would be a dry, theoretical husk of a subject. Imagine a music lesson where the students only studied the rules of counterpoint or the structure of sonata form, but never brought them to life by performing or composing such music!

But that's not all: programming is more than mere medium. As Fred Brooks put it, "the scientist builds in order to study, but the engineer studies in order to build". Most of the body of knowledge is organised around the challenge of building ever more ambitious edifices of software, and have them actually work and be useful. Programming is the very stuff of computer science.

Coding is phenomenally creative. The same Fred Brooks wrote that, "the programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realising grand conceptual structures."

When a child does a science experiment, she is seeing physical principles at work, coming to life in front of her eyes. But if she does the experiment right, we know what will happen. In contrast, when she writes a program, no one knows what will happen. The programmer brings into the world a

SIMON PEYTON JONES
Simon Peyton Jones FRS MAE is a British computer scientist who researches the implementation and applications of functional programming languages, particularly lazy functional programming.

new creation, formed from an infinitely malleable substance, which does something new, conjured from the mind of its creator. We are not limited by the strength of wood, or the budget of the school workshop: in programming we are limited only by our own (in)ability to manage the complexity of our creation.

Self-assessment and risks

Coding offers immediate, tangible feedback. No need to wait for the teacher to mark your English essay; in computing, the program runs, or not, and remorselessly exposes the logical errors in your thinking. When hunting a bug in a malfunctioning program, we form a hypothesis about what is wrong. We formulate tests that will confirm or refute that hypothesis. In the light of the results of those tests we refine the hypothesis, and so on – it is the scientific method in action. Even for students who will never explicitly 'program' again, programming teaches understanding and reasoning skills that are needed by everyone: business innovators (identifying the need/potential), scientists (working with data, developing computational models of scientific processes), of those procuring software (e.g. for the NHS, to know what is possible, what they should be looking for, what is good and what bad), or end users (because one must always have a notional machine model of what a piece of software is doing).

All that – but in addition, of course, programming is a tremendously useful and marketable skill. In every corner of business, and every part of our daily lives, there are programs, and they all need to be written, modified, fixed, and stitched together. There is tremendous demand for skilled programmers, who command high salaries as a result.

And yet, and yet. There are two risks here. First, the risk that we confuse the medium with the message. I fear a future prime minister giving a speech saying, "the new computing curriculum has been a great success: every child leaves school fluent in Python". What a disaster that would be! The computing curriculum is focused on ideas and principles, not on a particular technology like Python. Yes, some of those ideas (sequence, iteration, choice, abstraction) are directly embodied and brought to life in Python, but Python is just one embodiment among many, not the thing

> IN PROGRAMMING WE ARE LIMITED ONLY BY OUR OWN (IN)ABILITY TO MANAGE THE COMPLEXITY OF OUR CREATION

> ONCE PUPILS HAVE LEARNT TO CODE IN ONE LANGUAGE, THEY SHOULD BE ABLE TO QUICKLY TEACH THEMSELVES OTHERS

itself. Once pupils have learnt to code in one language, they should be able to quickly teach themselves others built on the same concepts, and also recognise those same concepts appearing in the wider world that surrounds them.

The second risk is that we may forget that the school computing curriculum is for the many, not the few. I certainly hope that the education our young people receive will inspire some of them to be the software developers of the future. But many more will become lawyers and plumbers, hairdressers and doctors. They all learn the elementary principles of natural science, and similarly should learn the elementary principles of computer science. And, just as mathematics appears in primary schools mainly in the guise of arithmetic, so computer science will appear mainly in the form of simple programming. Just as no one confuses arithmetic with the manifold glories of mathematics, so we should not confuse programming with computer science.

Life

Returning to Schleicher's blog post, he says, "the risk is that we will again be teaching students today's techniques to solve tomorrow's problems; by the time today's students graduate, these techniques might already be obsolete. We should instead focus on the computational thinking that underpins these techniques, and that students can use to shape the technologies of tomorrow."

Fair enough – and indeed computational thinking is already explicitly at the core of the English national curriculum from start to finish. But teaching programming is emphatically not "teaching today's techniques to solve tomorrow's problems". Programming is computational thinking incarnate, brought to life, made tangible, executable, and useful. It provides a powerful way to practise and so develop those computational thinking skills, and understand them deeply. People occasionally say, "in the future computers will program themselves", but I believe they are mistaken – we will simply increase the ambition of the programs we write.

So yes, to answer the question, we should teach our children to code. But we should do so not as an end in itself, but rather as a powerful and effective means to motivate, illuminate, and exemplify the underlying principles of computer science. There is no more intellectually exciting, creative, or practically useful subject. I want to convey to our young people a visceral sense of that richness and creative possibility, and by far the best way to do so is share with them the joy and beauty of programming. (HW)

WEBLINKS

Here's the link to Andreas Schleicher's blog post:

linkedin.com/pulse/should-schools-teach-coding-andreas-schleicher

Here's how *The Telegraph* interpreted it:

telegraph.co.uk/education/2019/02/21/teaching-children-coding-waste-time-oecd-chief-says

DIGITAL LEADERS IN THE PRIMARY PHASE

Children with a passion for technology can raise the profile of computing within the school community, whilst simultaneously developing their own leadership qualities

More and more schools are looking to the idea of recruiting Digital Leaders, whereby pupils can apply to take on such a role, in turn helping to develop their confidence and leadership skills. Digital Leaders can get involved in the use of technology within schools, and help support both teachers and pupils. We asked Kelly Frost to share her experiences, starting with a trip to make some news...

Visit to Sky Academy Studios

My students and I visited the Sky Academy Studios in order to make our own news report. Prior to our visit, we discussed the merits of a variety of topics, from bionic limbs to artificial intelligence, before deciding to look at whether screen time is damaging because of its universal relevance amongst children, parents, and teachers alike.

The children researched their ideas using the internet, before opting to have an expert witness (an imagined 'Professor of Sleep Studies'), an on-location report with a typical family, and an eyewitness report from a Head Teacher. We highlighted the importance of ensuring our sources of information were trustworthy, providing a real life context to the teaching of reading information with a critical eye in our digital literacy lessons.

We also discussed the need for our report to be balanced, so that it was valued as a reliable source of information without being biased. The children decided which role they would like to take on and these included being a director, producer, camera operative, and script editor, amongst others. Everybody felt they had an important role to play and that they were a significant member of the team. The children had a behind the scenes tour when they arrived at the studios, where they learnt about the latest innovations that Sky are pioneering, including developing a biodegradable alternative to plastic.

The children were engaged throughout the whole experience, and were well supported by a team of Sky experts to produce their own individual segments, complete with moving captions and voice-overs. The children came away with a first-hand understanding of the importance of teamwork, an appreciation for what goes into the making of television, and had the opportunity to use cutting-edge technology. They were so proud to watch the finished output, which was incredibly professional, and they erupted into spontaneous applause at the end! The children received a wristband with an in-built USB memory stick containing

a digital copy of the report as a keepsake to share with their families.

All those involved were left inspired, and one parent even relayed that their daughter now wants to pursue a role in the media after being a presenter in the project. The video was shared in classrooms during our digital literacy sessions, as well as via the school website to inform parents when considering their own/their child's access to devices. It really was an enriching experience for all those involved, and highlighted to the children participating how their interest in technology could support them in whichever career path they were to choose.

> “ THEY WERE SO PROUD TO WATCH THE FINISHED OUTPUT, WHICH WAS INCREDIBLY PROFESSIONAL

Application for *Blue Peter* badges

I consider it fundamental to encourage all girls to embrace the opportunities that computing presents, and they make up a significant presence within the Digital Leader team, with representation in each year group. In order to highlight the important role that women have played in the history of computing, we marked Ada Lovelace Day in October 2018, by compiling a digital fact file, consolidating our information technology skills on Purple Mash. The children then wrote an accompanying letter to *Blue Peter* to apply for a badge, detailing and enclosing their fact file about Ada Lovelace. The children were delighted to receive their badges by the post in due course, and have inspired many others to apply for one after sharing their achievement in assembly.

Website design for a local business

The children used the free app, Adobe Spark, to design their own website for a local newsagent without an online presence. We discussed the potential benefits of having a website for small businesses. Together the children devised a list of questions to ask the owner to compile the information that

BLUE PETER BADGES

Further information on how to apply can be found via the *Blue Peter* website: **bbc.in/2IKeqZQ**. The benefits include free access for children to over 200 attractions nationwide, until the age of 16. Children must be aged 6-15 to apply, and applications must be made on an individual basis. The application process can take at least ten weeks.

An excited Digital Leader having received his *Blue Peter* badge through the post!

© BBC

an effective website should include; such as opening hours, contact information, and services offered. They also had to source their own images, and decided on a layout for their site. We are passionate about giving children opportunities to be creative in using technology at our school, rather than just being passive users of it, which we hope was epitomised by this activity.

How were the Digital Leaders recruited?

An introduction to the role, the opportunities it presented, and the responsibilities it entailed was made in assembly, and interested children were encouraged to apply by completing an application form detailing why they love technology, what they access, what they hoped to gain from the role, and what skills they could offer to the team. An informal 'reference' was then sought from class teachers, before the team was decided upon. Most children opted to stay in the role after their initial term of one academic year, which allowed them the opportunity to develop their skills further, and created a real team ethos for our second year too. Other activities we have undertaken include visiting a local radio station and being interviewed for a live broadcast whilst there, creating a blog to introduce themselves, recording each other reading stories to share on the school website for the community to enjoy, and writing book reviews to display in the library for World Book Day. This is just a small snapshot of projects we have undertaken and I am very much led by the children's interests.

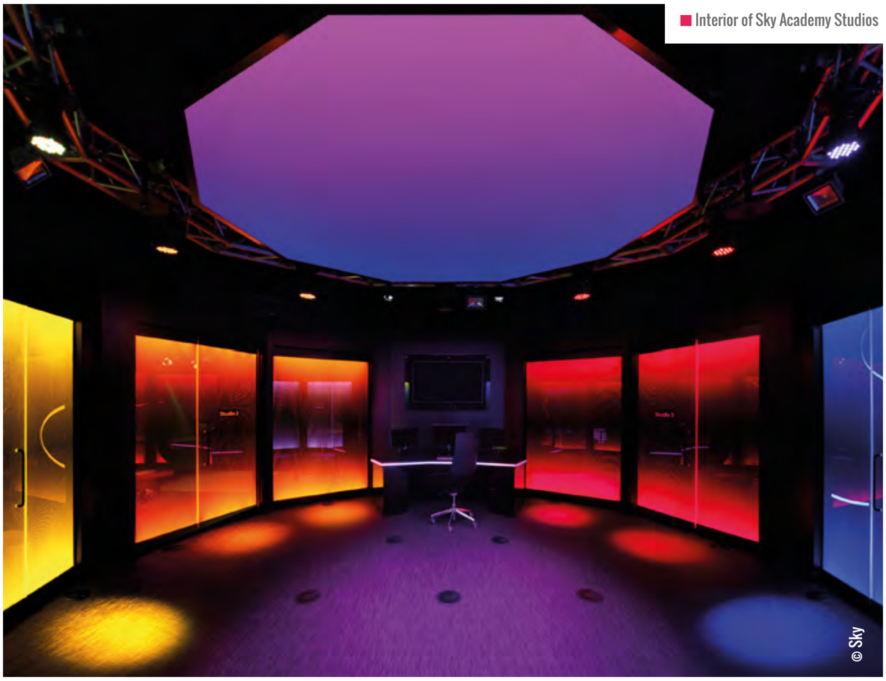
It is my hope that those involved have had the opportunity to develop a plethora of valuable skills during their time as Digital Leaders, which I hope have boosted their self-esteem in the immediate term and increased their employability in the longer term. Most pertinently, their self-confidence and leadership skills have been developed through the opportunity to support their peers and teachers in rolling out new software and hardware, as well as writing and presenting a segment for Safer Internet Day in assembly. Their enthusiasm and dedication to the role has made me very proud, and I look forward to continuing to work with them going forwards to see how technology inspires them, and the rest of our school, further in the future! (HW)

APPLY FOR A SKY VISIT

Register at **skyacademystudios.com** as a teacher to apply for a session. Sessions are released on a quarterly basis, and are free to attend. Suitable for 32 children, aged 7-18. Studios are situated in Osterley, West London and Livingstone, Scotland. Story trailers and promo videos are available as alternative activities to creating a news reel.

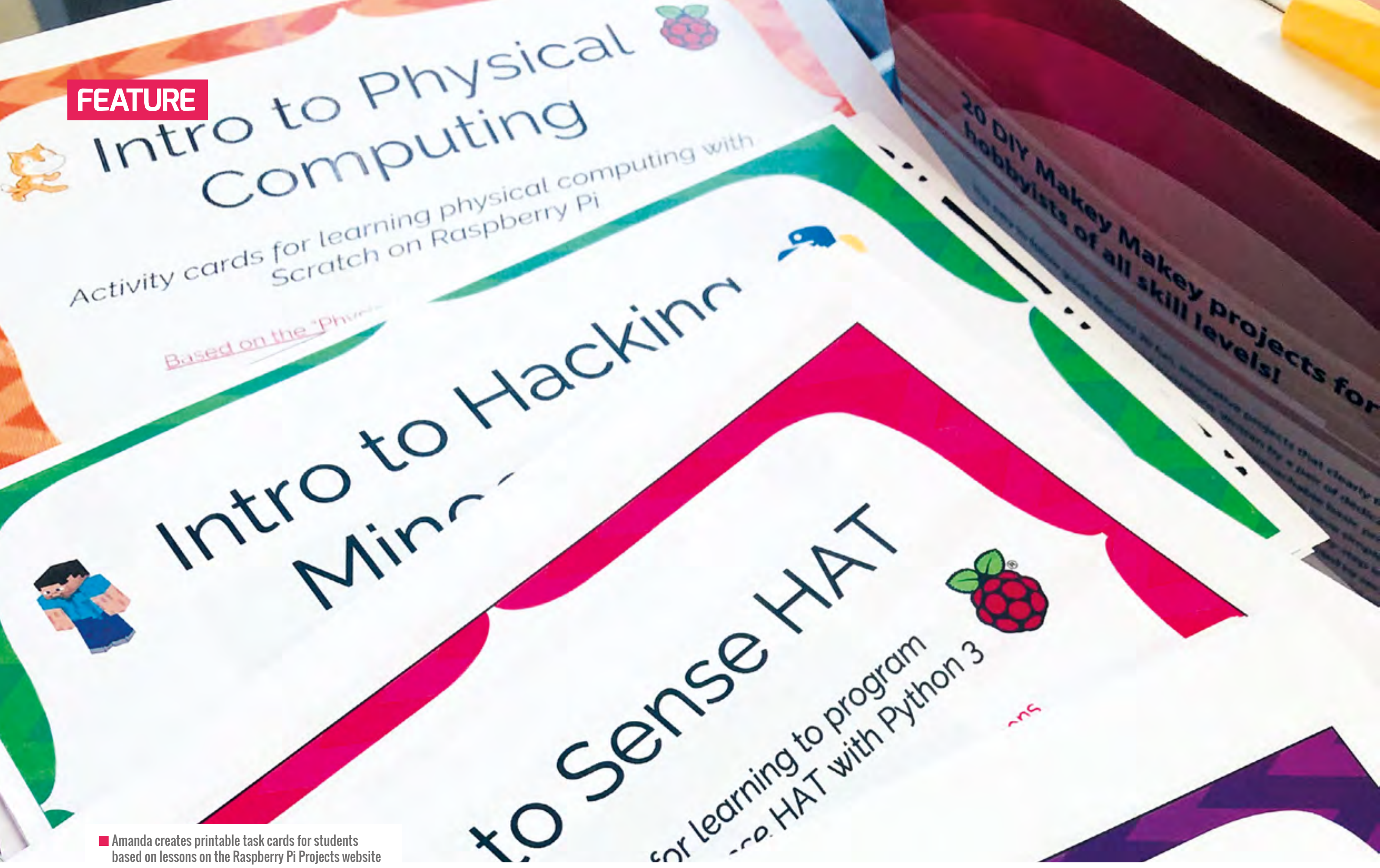
© Sky

> MOST PERTINENTLY, THEIR SELF-CONFIDENCE AND LEADERSHIP SKILLS HAVE BEEN DEVELOPED


Interior of Sky Academy Studios

© Sky

KELLY FROST
Y2 teacher and Computing Leader at St James Church of England Primary School, Weybridge, Surrey.


Intro to Physical Computing
Activity cards for learning physical computing with Scratch on Raspberry Pi
Intro to Hacking Min
to Sense HAT
for learning to program ... HAT with Python 3
20 DIY Makey Makey projects for hobbyists of all skill levels!

Amanda creates printable task cards for students based on lessons on the Raspberry Pi Projects website

PRIMARILY PI: TEACHING PHYSICAL COMPUTING IN K-5

With a little support, even our littlest learners can learn physical computing skills....

Educators are often hesitant to use physical computers with primary-aged students. But with a little scaffolding, even our youngest learners can learn programming and physical computing with something like the Raspberry Pi. Here are some tips for supporting primary-aged students through physical computing projects in the classroom.

Getting started

I teach in a school that does not currently have computer science-specific classes, so often my students come to me with different levels of experience in coding, or with no experience at all. For that reason, I usually start our physical computing work with a couple of offline lessons about computers and algorithms. This year, my favourite resource has been the *Hello Ruby* books. These stories about a little girl named Ruby, who gets into all sorts of adventures inside her computer, are a great introduction for young students to hardware, software, coding, and the internet. The *Hello Ruby* website (**helloruby.com**) also has all sorts of offline activities, geared towards young students, for practicing computing terminology, thinking logically, writing algorithms, and more.

Once students have been introduced offline to vocabulary and concepts they'll need in our digital making project, we use **code.org**'s self-paced lessons to introduce block-based coding languages, and how to write a sequence of code with blocks. Students work through a handful of lessons to get a grasp of the basics of block-based coding:

- Dragging and dropping blocks into a work space
- Connecting the blocks
- Writing a sequence of directions from top to bottom
- How to trash blocks when you make a change
- How to run your program
- Analysing for errors and debugging

The **code.org** intro lessons also help students practice basic computing skills that

A group of 2nd graders celebrating coding success while working on their urban wildlife cameras

> MOST OF OUR PRIMARY STUDENTS ARE MORE FAMILIAR WITH TOUCH SCREEN AND TABLETS THAN WITH COMPUTERS

many don't have coming into school. Most of our primary students are more familiar with touch screens and tablets than they are with computers, so we also need to teach:

- How to use a mouse or trackpad
- How to click and drag with a mouse or on a trackpad
- How to 'right click' on a trackpad or mouse
- How to navigate software menus and toolbars

Learning by making

After just a few lessons in **code.org**, we dive right into our work in Scratch and physical computing. One of the things that I loved about Picademy was the project-based approach, so I decided to adapt that format for my own students. Learning within a relevant context in any subject area tends to be more motivating for my students than trying to learn skills in a siloed approach, so we start by introducing what we'll be making. Then we teach the individual coding and electronics skills necessary in order to get there.

I tend to integrate our physical computing projects within units currently being taught, and I design the lessons with a cross-curricular approach. Some of my favourite projects have included:

- Learning about traffic lights with our Kinder 'traffic engineers' (community jobs unit)
- Writing gold-finding programs in Minecraft with our 4th graders (California Gold Rush unit)
- Designing urban wildlife cameras with my 2nd graders (while learning about ecosystems in their habitat-themed literacy unit)
- Digital voting booths with both 2nd and 3rd graders (government units)

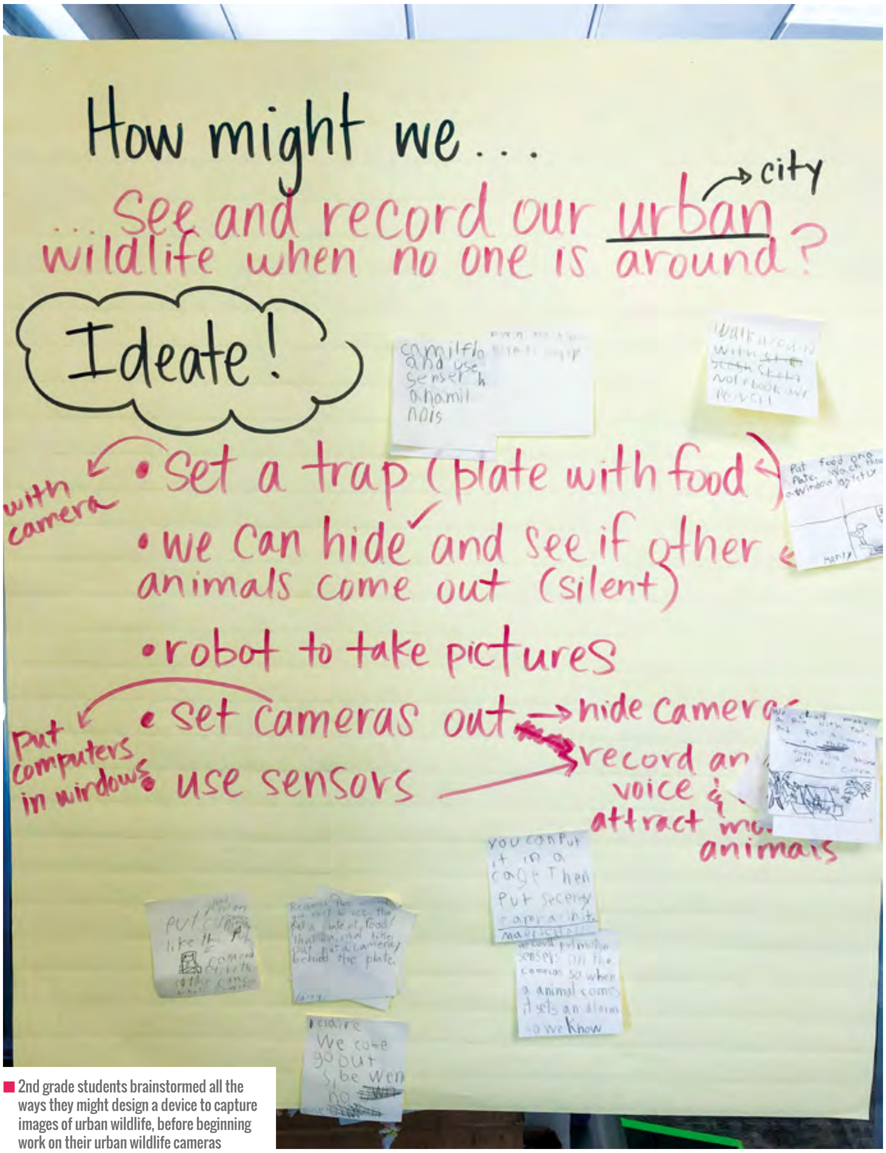
I like to embed opportunities for students to think like designers within the project. Even if I already have a plan for how we're going to make something, we launch with a student-led brainstorm, asking students what elements our digital making projects should include and why. What might we need to create an effective wildlife camera? How should we design our digital voting booth to make voting an engaging process? If needed, I ask guiding questions to help them come up with any 'missing pieces' in our plan. "So far you've all decided that our wildlife cameras need the camera lens, and no flash because we don't want to scare the animals, but I'm still not sure how our camera will know that it's time to take a picture… what else will it need?… Oh, Henry, great idea… cameras usually have a button. Okay, let's add that to our plan."

Scaffolding electronics

When working with young students, hooking up the electronics (LEDs, jumper cables, PiCameras, motors, etc.) can sometimes become frustrating and slow down their work, especially during their first lessons on a Pi. At the K-1 level, I do all the electronics set up for the students in advance.

AMANDA HAUGHS

Amanda Haughs is a 2nd grade educator and learning designer at the Campbell School of Innovation in Campbell Union School District (San Jose), California. She previously worked as an instructional coach supporting students in grades TK-8, and in her current position supports students in grades 2-4. Her focus is on engaging all students in meaningful and authentic cross-curricular learning experiences, with the use of technology to personalise and enhance learning. Haughs is passionate about computer science education and STEAM education in elementary classrooms. As a 2015 PBS lead digital innovator, 2016 Silicon Valley CUE outstanding educator, Raspberry Pi certified educator, Apple teacher, Google certified educator, and Leading Edge certified Professional Learning Leader, she works to develop creativity, critical thinking, and problem-solving skills in students via digital making, design thinking opportunities, and multi-disciplinary, project-based instruction.


How might we...
city
See and record our urban wildlife when no one is around?
Ideate!
camilflo and use senser h a hamil nois
walk around with ~~stch~~ ~~stch~~ skta notebook and pencil
with camera
• Set a trap (plate with food)
Put food and Plate. Wach thru a window asyely
• we can hide and see if other animals come out (silent)
• robot to take pictures
Put computers in windows
• set cameras out → hide cameras
record an voice & attract more animals
• use sensors
you can put it in a cage Then put secerty camra in it.
we could put motion sensers on the cameras so when a animal comes it sets an alarm so we know
We could make a box with ... put a camra ...
Regrand the ... put a plate of food that animal like put a camera behind the plate.
We cute go out and be wen no

2nd grade students brainstormed all the ways they might design a device to capture images of urban wildlife, before beginning work on their urban wildlife cameras

During a mini-lesson, students learn next steps in their program and then return to their stations with their teams to try what they've just learned

For my 2nd-4th grade students, I usually set up the electronics for them prior to the first couple of lessons, but as we progress through the project, I release a little more of the electronics set up each time we continue on a new lesson. Diagrams and pre-breadboarded models allow students to learn some basic breadboarding skills by copying examples.

One of my favourite add-ons for digital making with young students is the Pibrella HAT by Cyntech and Pimoroni. After several days of modelling, our young students learn pretty quickly how to properly put HATs on the Raspberry Pi. And with LEDs, buzzers, and buttons already built into the board, and the ability to program the HAT in Scratch, it's a great alternative to breadboarding. (Plus the inputs/outputs on the Pibrella HATs make it easy to also program motors, sensors, and more, so this versatile HAT has become one of the most used tools in our electronics kits!)

The Sense HAT add-on is also a favourite among my primary programmers and is easy for young students to code using Scratch or Scratch 2.0. They love lighting up LEDs and 'drawing' pictures on the Sense HAT with just a few blocks.

Another way that I scaffold the electronics work for my young students is by setting up breadboards for them ahead of time. I organise the LEDs, buttons, resistors, and jumper cables on the breadboard, and then walk students through how to connect the jumper cables to the GPIO pins. As they become more comfortable with using and programming on their Pis, then we move onto circuitry and get students setting up their own breadboards or plugging in their own HATs and PiCameras.

Getting coding

One of the (many) reasons I'm keen on the Raspberry Pi is the ability for students to program the physical world using Scratch. The ease with which students learn Scratch makes the introduction to physical computing that much more approachable.

Using Scratch does require some amount of reading skill, however. The colour-coded blocks help, but depending on the grade level and reading skills of our students, I sometimes start with an explicit vocabulary lesson to help students with language they'll need while coding. Just as we would in other

2nd grade students work in teams of 3-4 on their physical computing projects, helping each other learn and troubleshoot

subject areas, we might display the blocks we'll be using on a chart and practice reading them together, using physical cues and images or sketches to help students learn the words.

Sometimes we print out the Scratch blocks on paper (available on the Scratch Ed website) and practice reading them and putting them into the sequence we want ahead of time before we work on the computers. We sometimes have them act out the code, as well, so they can test whether the program they've planned will move their sprite the way that they want. Students can then use the plan they made on paper as a resource for finding the blocks they need once they get started in Scratch.

Other times, I've loaded a pre-written Scratch program onto the students' computers, with the blocks they need already in the scripting area, but out of order. Students decide what order the blocks need to go in and then snap them together to create their program.

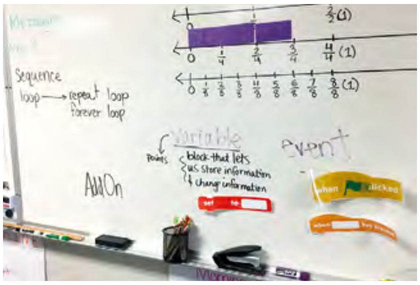
I'm also a fan of the 'learning by copying' method of learning to code. For all projects I've done with my classes, I've created activity cards that walk students through each step of the project so they can copy (and eventually, customise, when they're ready) the code that they need, line by line. The cards include large visuals and diagrams, along with the code students can copy to create their own projects. I often will include pop-outs in the diagrams that explain in a couple of words what certain blocks or lines of code mean or do, so students can learn programming and computer science concepts while making.

The activity cards allow students and teams to move at their own pace on a project, and the ability to check and debug their work against an already-working program. The task cards have also become a great way for me to teach students how to read technical texts and follow a set of given directions in order to put something together, and a way for students to work on projects on their own time (not just at times that I'm leading the lesson for them).

Student becomes the teacher

I like to group my primary-aged students into teams of three when working on a physical computing project. We try to balance each team with students who feel that they are most skilled or most comfortable in each of the following areas:

- Coder/typer
- Debugger/editor
- Engineer/electrician



Before programming their computers, students use printed Scratch blocks to practice and plan the program they will write in Scratch

Sometimes I create the teams ahead of time, but sometimes I ask students to think about the skill they feel most comfortable in and then to build and balance their own teams.

While everyone on the team will get a chance to participate in all parts of the project, we tell students that if they feel they are the strongest 'coder' on the team, they can be the coding coach on their team. The strongest 'editors' will be the coach in that area, and while we expect everyone on the team to take time to check spelling, capitalisation, spacing, etc in the code, the editor will lead the process. The 'engineers/electricians' are the students who feel most comfortable with setting up the hardware and sometimes, setting up circuits.

This team format is a great way for us to explicitly teach students collaboration skills. Primary-aged students aren't always naturally skilled at making sure everyone gets a turn, so, before students are allowed to get to work on projects, we discuss as a class what strategies we'll use to make sure that each person on the team is able to participate. I often set up the expectation that each person will take a turn dragging a block into the code and then pass the mouse to the next person, continuing the rotation until the entire program is written.

As for direct instruction skills lessons, I also try to keep my students moving frequently. I usually break my physical computing lessons into sets of mini-lessons. Students come to the carpet for a mini-lesson, then go back to stations to work for a bit, then I call them back for next steps, and then send them back to stations to work. Chunking the lessons gives students smaller benchmarks to work towards, and more manageable amounts of information to try and remember at one time. And bringing them to carpet ensures they're focused on the new lesson and less tempted to continue working while I'm giving new directions.

As they learn new skills, student 'experts' start to emerge, one of the best benefits of the team approach to physical computing in our classroom. I don't have to be the only helper in the room. As some students start to become more and more confident in their programming skills, they become the helpers in the room, not only for their own team, but also for others in our class and, sometimes, for me as well. (HW)

MODELLING THE GLOBAL CARBON CYCLE

Will Grey explains how we use computers to help us model environmental systems

We can use computers to help us simulate the physical world. One area where computer simulations are used extensively is in climate and weather forecasting. Weather models allow us to forecast the weather a few days from now, and climate models allow us to predict future temperatures and other climate variables ten, 50, and even hundreds of years ahead. This is a remarkable achievement that has been accomplished by applying our understanding of physics and maths, and formulating them as algorithms that are run on the most powerful supercomputers in the world.

Computer models are abstract representations of reality. We take the most salient aspects of the Earth's weather and climate system and represent them mathematically as computer algorithms. The less relevant aspects we ignore. Which aspects of nature we consider important of course depend on where we stand. For instance, in the early days of climate modelling, the land processes were ignored but now they are integral to climate models.

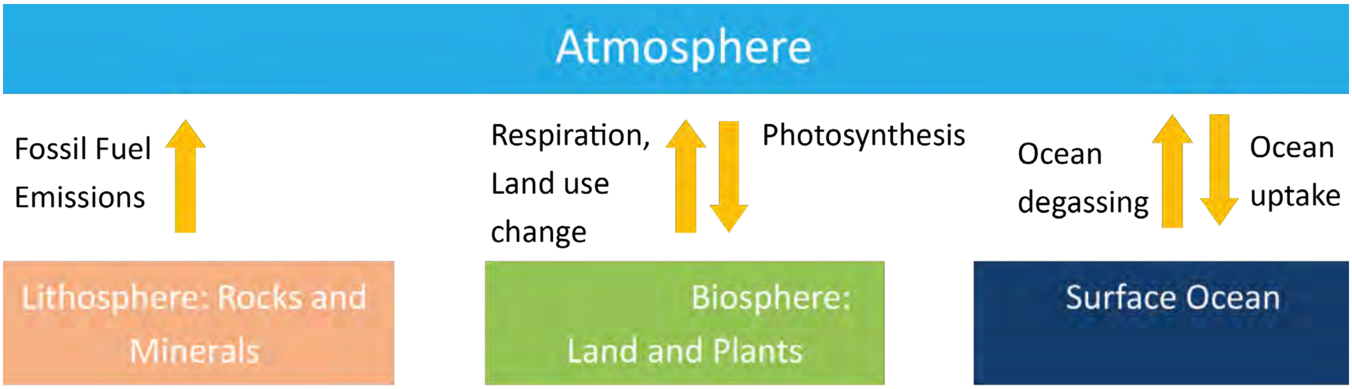
But how do we know whether our models are any good? In the field of climate and weather, we have a mind-bogglingly vast volume of observations. These observations come from a worldwide network of weather stations on the land and ocean, and from weather satellites orbiting the Earth. The model outputs can be compared with these observations. Where the model and observation match closely we can assume that the model performs well. Where there is a less good match, then we can investigate further and make improvements as required.

Climate change is the issue of our times and computers are playing a critical role in helping us learn about this extremely complex system. As we know, carbon dioxide is an important greenhouse gas that plays a key role in regulating the Earth's temperature. Current levels of atmospheric carbon dioxide are at their highest level in at least the last 800,000 years because of human activity.

In this article, we will explore a simple global carbon model that we will use to explore this issue. Despite its simplicity this is a very powerful tool in helping us estimate atmospheric concentrations of carbon dioxide.

EMISSION SCENARIOS

- A1 - Rapid economic growth, population peaks 2050, introduction of efficient technologies
- A2 - Continuously growing population, slow introduction of efficient technologies
- B1 - Population peaks 2050, but rapid adoption of efficient technologies
- B2 - Continuously growing population, but regional development

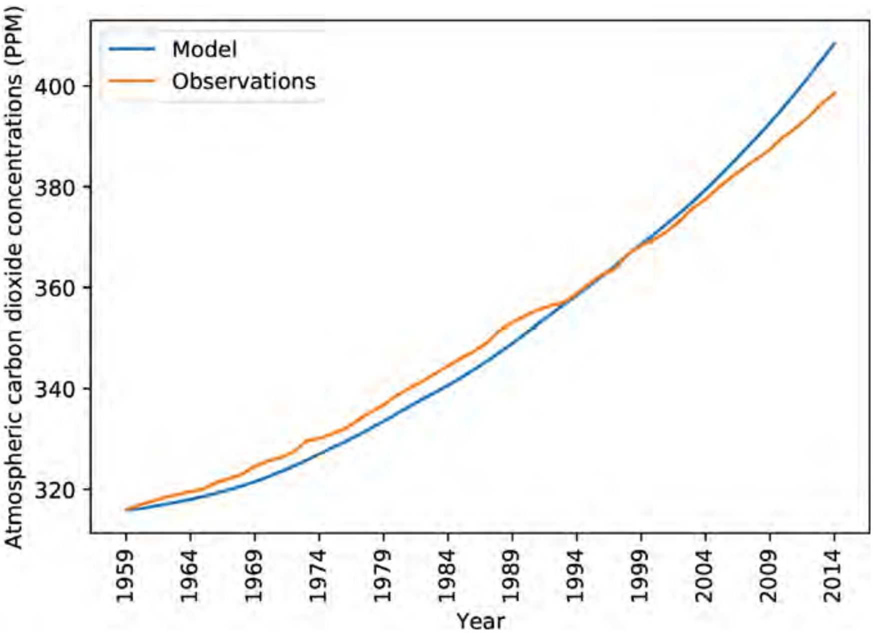

Atmosphere
Fossil Fuel Emissions
Respiration, Land use change
Photosynthesis
Ocean degassing
Ocean uptake
Lithosphere: Rocks and Minerals
Biosphere: Land and Plants
Surface Ocean

The carbon store box model showing the reservoirs and fluxes

Carbon model

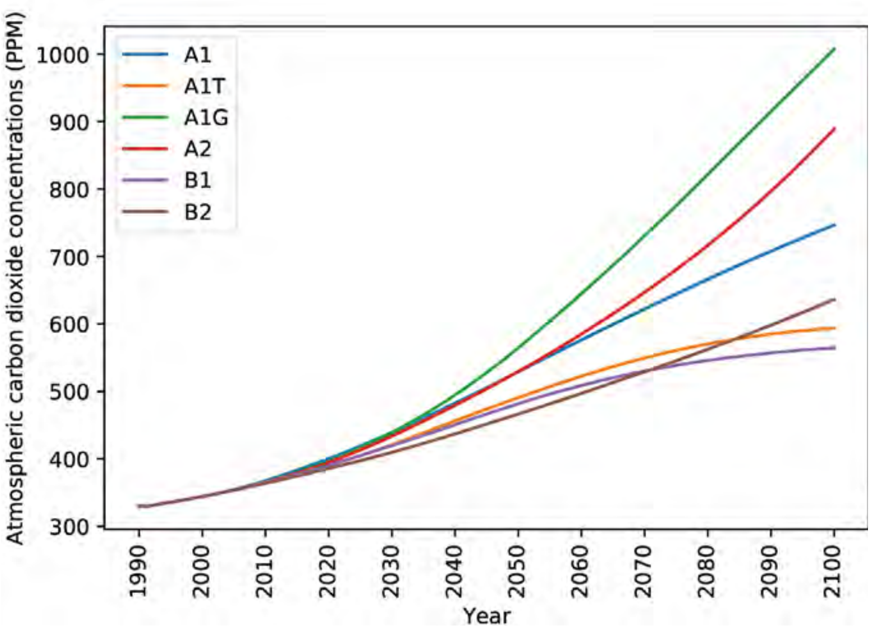
Here we are using a box model that consists of reservoirs, where carbon is stored for different components of the Earth's climate system, and fluxes which represent the transfer of carbon between those reservoirs.

The reservoirs in our model are the atmosphere R_a, the surface ocean R_o, the biosphere (land and plants) R_b, and the lithosphere (rocks and minerals) R_l. We assume a closed carbon system, where carbon cannot be created or destroyed. The fluxes between the different components of the Earth are both natural and anthropogenic. Atmospheric carbon dioxide can be dissolved naturally in the ocean F_u, and in the other direction, ocean degassing F_d can transfer carbon dioxide from the ocean to the atmosphere. On-land photosynthesis F_p – where carbon is taken up by plants – and respiration F_r are natural processes. However, land cultivation F_l by humans is having a net cumulative effect of reducing carbon uptake by plants. Nowhere is this more evident than in the Brazilian Amazon, where the primary rainforest is being replaced by agricultural land at an alarming rate. Finally, the burning of fossil fuels F_f means that carbon that has been stored in the ground for millions of years is being released into the atmosphere. This represents the transfer of stores of carbon from the lithosphere (e.g. oil, gas, and coal) to the atmosphere.



■ Atmospheric concentrations of carbon dioxide as calculated by the model and shown against observations for comparison

The model has an initial quantity of carbon in each of the reservoirs but this changes at each time step. The fluxes represent transfer of carbon over a period of a year. At each year-long time step, we add and subtract the fluxes of carbon to and from each reservoir and update the store of carbon in each of the reservoirs.



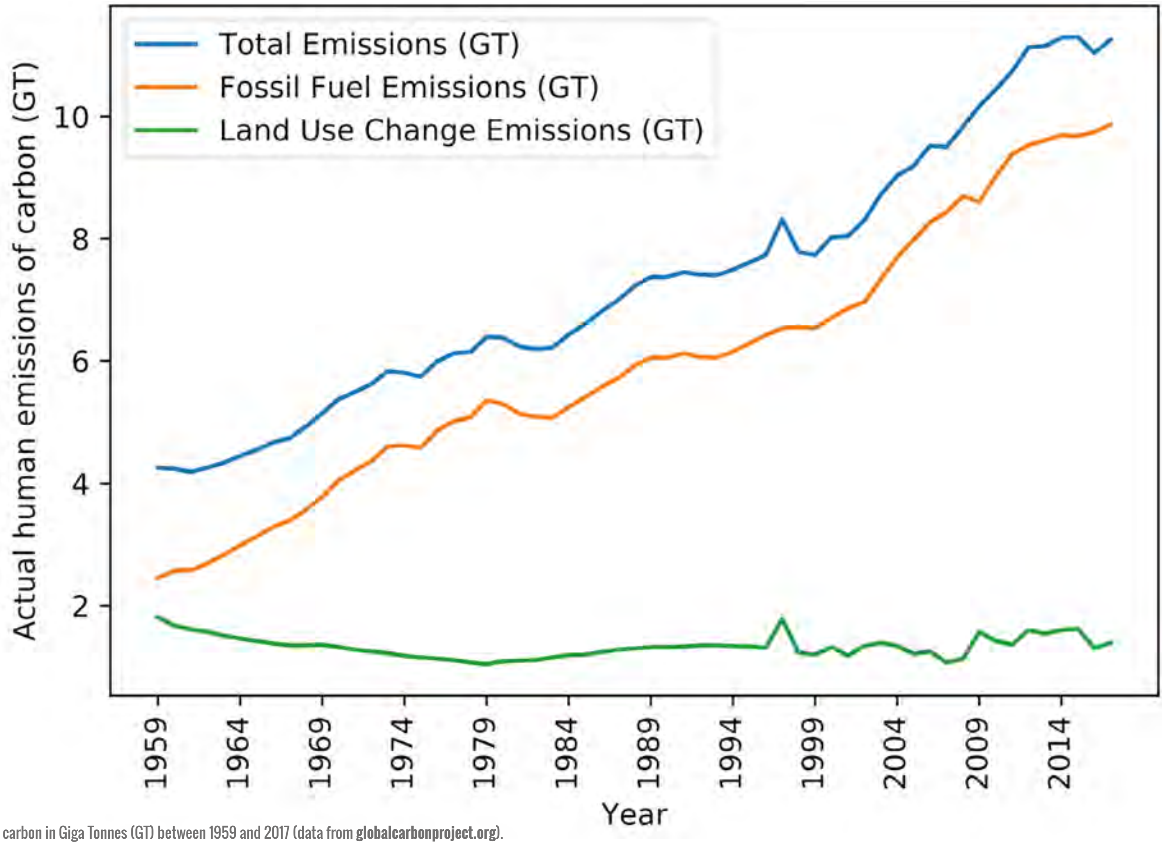
■ Atmospheric concentrations of carbon dioxide projected for six different emission scenarios.

To run the model over several years, we repeat this update for each time step. At each time step, approximately half of the carbon emitted by human activities remains in the atmosphere and the other half is absorbed by the biosphere and oceans.

Now that we have conceptualised our model, the next stage is to formulate it into a mathematical framework and convert it into an algorithm so that it can be automated as a computer program. First, we define the quantity of carbon in each of the reservoirs. We do not need a quantity for R_l because it is not used in any of our calculations. The initial quantities in each of the reservoirs are as follows:

$$R_a=2.1\times 315.97\ GT$$
$$R_o=1000\ GT$$
$$R_b=3000\ GT$$

By convention, we present atmospheric carbon dioxide concentrations in parts per million (PPM). To convert between parts per million (PPM) and gigatons (GT), we multiply by a factor of 2.1. In our first run of the model, we start in 1959 because this is when atmospheric concentrations of carbon dioxide started being recorded. ▶



Human emissions of carbon in Giga Tonnes (GT) between 1959 and 2017 (data from **globalcarbonproject.org**).

In 1959, there were 315.97 parts per million molecules of carbon dioxide in the atmosphere.

In Python, the corresponding assignment is coded up as:

```
atmosStore =315.97 * 2.1
surfaceOceanStore = 1000.0
biosphereStore = 3000.0
```

Next, we define the fluxes that represent the amount of carbon transferred between the reservoirs per time step. The coefficients for natural fluxes of carbon have been taken from Rodhe. The model requires input data of human carbon emissions for each year over the duration of the model run. For testing, our model F_f and F_l are taken from the global carbon project for the years 1959 to 2017 (see the chart at the top of this page). The other fluxes are given by:

$$F_u = 0.15 \times R_a$$
$$F_d = 1 \times 10^{-25} \times R_o^{9}$$
$$F_p = 16.4 \times R_a^{0.2}$$
$$F_r = 0.019 \times R_b$$

The corresponding Python code for the fluxes are:

```
oceanUptakeFlux = atmosStore
* 0.15
oceanDegassingFlux = 1e-25 *
oceanStore**9.0
photosynthesisFlux = 16.4  *
atmosStore**0.2
respirationFlux = 0.019 *
biosphereStore
```

The change in carbon stored in the reservoir per time step is given by the following set of differential equations:

$$\delta R_a = (F_f + F_l + F_r + F_d - F_p - F_u)\ \delta t$$
$$\delta R_b = (F_p - F_l - F_r)\ \delta t$$
$$\delta R_o = (F_u - F_d)\ \delta t$$

We can ignore δt because $\delta t = 1$. That is to say our time step is 1 year. Therefore after each time step we have:

$$R_a = (t+1) = R_a(t) + \delta R_a$$
$$R_b = (t+1) = R_b(t) + \delta R_b$$
$$R_o = (t+1) = R_o(t) + \delta R_o$$

The store of carbon at the next time step at $t+1$, is being updated by adding the net change in carbon to the store of carbon at the current time step. The corresponding Python code is given as:

```
for i in range(years):
```

```
 atmosStore = atmosCarbonStore
+ respirationFlux +
oceanDegassingFlux +
human_emissions[i] -
photosynthesisFlux -
oceanUptakeFlux
```

```
 oceanStore = oceanStore
+ oceanUptakeFlux -
oceanDegassingFlux
```

```
 biosphereStore =
biosphereStore +
photosynthesisFlux -
respirationFlux
```

```
 oceanUptakeFlux = atmosStore
* 0.15
 oceanDegassingFlux = 1e-25 *
```

```
oceanStore**9.0
 photosynthesisFlux = 16.4  *
atmosStore**0.2
 respirationFlux = 0.019 *
biosphereStore
```

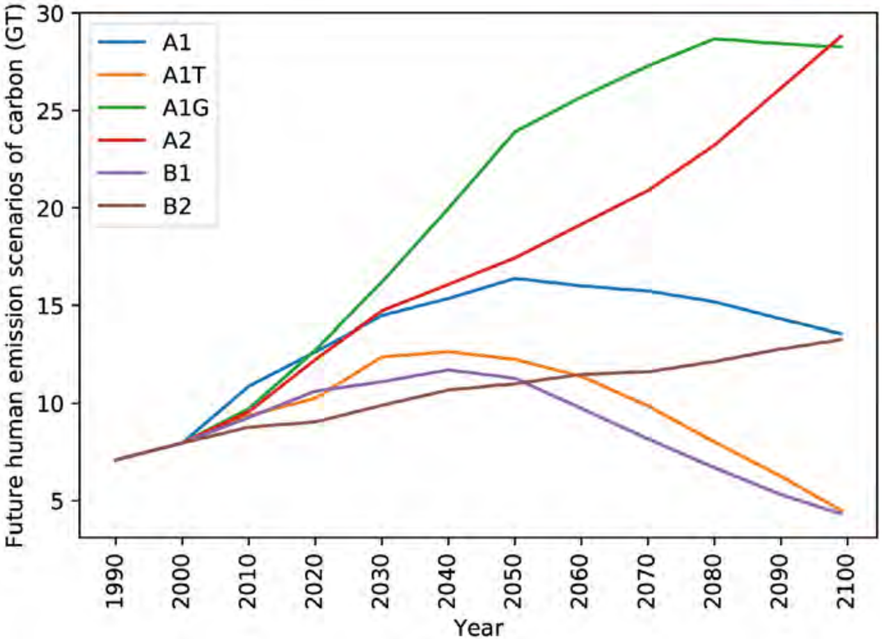
Remember, at each time step we also have to update the fluxes because the stores on which they are calculated have also changed.

Running the model

The model was run for the period 1959 to 2014 to see how accurately we could estimate the atmospheric carbon concentrations. For this period, we also have observations of atmospheric concentrations at Mauna Loa in Hawaii to compare our model against. The results are shown in the chart at the top of page 73, and we can see that we produce a decent representation of the observations, considering that this is a pretty simple model.

We can be confident that we have a reliable model which we can use to predict future atmospheric carbon concentrations. This is where the power of the modelling approach comes into its own because there is nothing else that allows us to do this.

For our model, we need to input data on human carbon emissions. Of course, we do not know what future carbon emissions are going to be, however the intergovernmental panel on climate change (IPCC) has identified a range of possible emission scenarios up to the year 2100 that depends on global human population, and how quickly we adapt to sustainable ways of living. To be clear, these are a range of conceivable scenarios, not actual emissions.



Projected human emissions of carbon in GT for six emission scenarios. (Data from **ipcc.ch/report/emissions-scenarios**)

The IPCC has identified four families of scenarios representing a range of storylines (see boxout). These scenarios are shown in the chart at the top of this page. We ran the model using each of these scenarios in turn to determine what atmospheric concentrations of carbon dioxide will be in the future. The output is shown in the chart at the bottom of page 73. We have no idea if any of these realities will play out, but it give us some clue as to what can happen. For instance, if we continue business as usual along the A2 scenario, in the chart above, it is clear that atmospheric concentrations of carbon dioxide will continue to grow at faster and faster rates over the next century. However, if we do manage to adapt to more sustainable lifestyles, atmospheric carbon concentrations can be managed.

Although computer models and simulations are not in the current A-level specification, I do like to present this model to my A-level cohorts to give them a wider exposure to a mathematical application of computer science that they would not otherwise get. This also serves as a starting point for an A-level project and there are lots of directions in which we could take this work. For instance, we could run the model at monthly time steps so we could see what happens to atmospheric concentrations in different seasons. We could use the Kaya identity (**en.wikipedia.org/wiki/Kaya_identity**) to develop our own emission scenarios. We could carry out analysis on a country by country basis instead of globally. We could even look at other time integrations models such as Lovelock's Daisyworld (**en.wikipedia.org/wiki/Daisyworld**). (HW)

SOURCES OF INFORMATION

- Rodhe, H. and Bjorkstrom, A.
  Some consequences of non-proportionality between fluxes and reservoir contents in natural systems
  Tellus (1979), 31, 269-278.
  bit.ly/2ktGbpf
- Carbon dioxide concentrations from Mauna Loa observatory record: **bit.ly/2kFR5r0**
- Future emission scenarios from: **bit.ly/2kwkDs8**
- Emission data from: **bit.ly/2kdlbBP**
- The full code listing is here: **bit.ly/2kysc1s**

WILL GREY

Will Grey is head of computing at Comberton Village College, a coordinator of the CAS Cambridge Secondary Community of Practice, and a former climate and Earth observation scientist. He has developed a comprehensive set of resources for A-level and GCSE computer science at **helloworld.cc/grey**.

AGE RANGE

UK year 5 and 6

RESOURCE TYPE

Exploratory Learning

REQUIREMENTS

Computer with Logo programming language such as:

- Windows computer with MSWLogo installed (**mswlogo.en.softonic.com**)
- Internet access and **turtleacademy.com/playground** unblocked
- iPad with Logotacular installed

SHAPE EXPLORER

Using programming to explore angles and shapes in an interactive and creative way

Knowing things, and being able to use that knowledge, are two very different skill sets. As teachers, we often try to provide meaningful ways to encourage children to problem-solve and apply their learning and, using this lesson, I found the children had more conversations about angles than in my week of angle-related lessons previously.

Using a simple text-based programming language, the students are set the task of drawing regular polygons using what they know about 2D shapes. How many sides should it have? Does it have right angles? Are the other angles acute or obtuse? Do you need those angles to be bigger or smaller than they are? A few simple commands and they're off exploring!

The key to this lesson is starting off simple, and then getting progressively more challenging. It doesn't take long to draw a square, as long as you know a right angle is 90 degrees. A pentagon, however, takes a bit more time! The beauty of this exploration is it can easily be adapted to suit everyone's needs. Some students will choose a trial and error approach, others will look for patterns, some will sketch the shape on paper first, others will type instructions straight away, but all students will get immediate feedback.

The main content of the lesson plan follows below. Feel free to repeat or delete sections as relevant (e.g. some lessons may have two main activities). (HW)

ACTIVITY 1: HOW DOES IT DO THAT? 5 minutes

fd 200
fd200
df 200
forward 200

I now show the children a few simple drawings and explain that I programmed the computer to create them. With their partners, they must work out which program created which picture (and which did not do any). It's important to note that, at this time, I don't explain what 'fd' means and the children have not used Logo before. I just want them to use their reasoning and deductive skills to have a go.

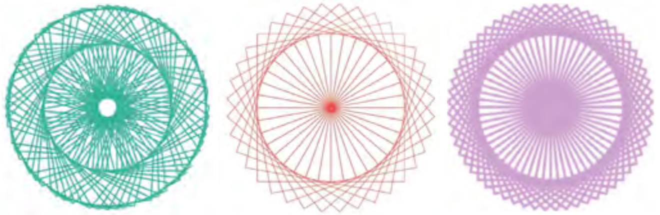
After two minutes, bring the children back together, but don't focus on which is which, instead focus on the code. Did anyone make a guess as to what 'fd' meant? Talk them through the code and tease out it's fd for forward, rt and lt for right and left. Now, what about 90? As we've been covering angles all week, this shouldn't be too difficult and then we reach 200... what's that? The answer is pixels. It doesn't really matter if they know this, but lots of children write 'fd 1' and think nothing has happened. As one pixel is so small, they just don't see it, so it's worth drawing their attention to that here.

Next, we address some basic misconceptions that are likely to get in the way of the maths learning! Here there is just an image of a line, but several code options – such as fd200, fd 200, df 200, forward 200 – and the children must explain which ones will and won't draw the line and why. This focuses the children's attention on the importance of careful typing, spacing, and that the nuances of whichever version of Logo you are using.

LEARNING OBJECTIVES

✓ To explore the angles of regular polygons

✓ To use text-based programming to draw shapes



Some of the simple drawings, where the children try to work out which program created them.

ACTIVITY 2: USING WHAT WE KNOW ABOUT REGULAR SHAPES 30 minutes

Now is the time for the students to get hands-on with Logo. Show them how to access it (program, app, or website) and check they can all confidently make a line using one of the examples from the starter. Then, offer them a challenge. Who can draw a square? Encourage them to think about the properties of the shape first, they can even draw it if that helps them to visualise what the computer needs to be told to do.

When someone achieves the square, this is the time for a mini plenary to draw out the maths. What did they do? How did they think about it? Talk about how a square has four equal sides, so each side would have the same length. To draw a side, they used the 'fd' command. What about the angles? What do they already know about the angles on a square? How did this help them with the turns? Did every corner need a 90 degree turn? Why? Maybe there's a pattern there.

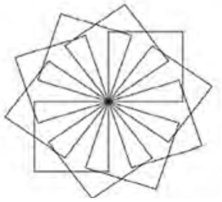
Now the challenge is to keep making shapes. What about a pentagon? Hexagon? Octagon? Circle? Encourage the children to keep notes about which angles they used for which shapes, as this will be useful for them to start noticing a pattern. What were the angles on a square? 90? Oh, what about the pentagon? 72? Interesting, so it got smaller from a four-sided shape to a five-sided shape. What do you think will happen for a hexagon?

Whilst interacting with the children, encourage them to use the mathematical language related to angles. Is that angle you've made acute or obtuse? Is that a reflex angle? Can you remember how many degrees it is to turn all the way around? All of these questions will help the children explore creating shapes using Logo.

After all the children have created at least three shapes, or 20 minutes of activity time has passed, pause the lesson and ask the children to look at what angles they've found so far. Can they spot any patterns? Give them several minutes to discuss this with their partner before sharing with the class. Then ask them to tell you how many degrees are in a full circle. Can anyone work out 360 divided by four (the number of sides on a square?) 90! Where have we heard that before? What about 360 divided by five?

Repeat until the pattern is clear, and then give the children five more minutes to see what shapes they can make with this formula. It's also a good way to get them to practice their division skills!

ACTIVITY 3: ART? MATHS? OR COMPUTING? SHAPES 15 minutes



To conclude the lesson, I like to bring together the use of Maths and Computing and add in an artistic element. This time I show children the image of a pattern and ask them to think about how I made it. Talking to their partners they should begin to recognise that this is a series of squares. By turning after creating each square it makes a pattern.

Show the children the code used to create the pattern:

repeat 10 [repeat 4 [fd 100 rt 90] rt 36]

Highlight the fact that the code in the middle is a shortened way (ideally, link to their use of repetition in programming from computing lessons) of drawing a square. Can the children explain why I've drawn ten squares and turned 36 degrees between each one? If necessary, show them what happens if you only draw eight squares and it fails to complete the pattern.

Finally, conclude that the number of squares drawn multiplied by the angle turned equal 360 degrees, so the circle is complete before giving the children five minutes to explore the patterns they can make. They can copy the code provided and change the number of squares and degrees turned to factor pairs of 360.

SWAY GRANTHAM

Sway (@SwayGrantham) is a Computing Teacher at Giffard Park Primary School in Milton Keynes and Primary Curriculum Manager for the National Centre for Computing Education. She is a Raspberry Pi Certified Educator and a Specialist Leader in Education.

ASSESSMENT

Mini plenaries are a chance for you to question students' understanding of what they are doing and how they are progressing.

DIFFERENTIATION

As the lesson is exploratory, it is accessible for the majority of learners. If there are some who are unable to work in this way, the code to make a square can be provided but with missing angles, thus reducing the cognitive load expected, whilst allowing them to meet the learning intention.

For those who need further exploratory tasks, or recognise the pattern earlier on in the lesson, giving them the challenge of creating a circle works well. Whilst a curved line is not technically possible, moving one pixel and then turning one degree 360 times gives a good illusion and makes them think about how angles work.

AGE RANGE

11-13

LESSON TYPE

Programming
Science enquiry and investigation
Cross-curricular Science activity

REQUIREMENTS

- Raspberry Pi with SenseHat or SenseHat emulator
- Sense HAT emulatorontrinket
- Python 3

CRASH-LANDING - PART ONE

Spencer Organ survives a crash landing in the jungle using a Raspberry Pi and Python. Introducing the SenseHat..

You were travelling on holiday with your trusty bag of technology including a Raspberry Pi, SenseHat, and micro:bits. Half-way there, you crash-land in the jungle. You will need to use your wits, cunning, and programming skills to survive your time in the jungle.

No bugs will get eaten.

These activities can also be carried out as an independent project, or in a club setting.

Code the Sense HAT in Your Browser!

If you don't have access to a SenseHAT, you can use the SenseHAT emulator on the Raspberry Pi. You can also use the trinket SenseHAT emulator here: **trinket.io/sense-hat**

The challenge

In this first part – and we'll have part two next issue – we will learn how to take some environmental measurements, and then use them to respond to the readings.

- Measure the temperature of the surroundings
- Measure the air pressure and detect if it is changing
- Predict if there is a storm coming

The activities

Our first two tasks help us explore the SenseHAT and start to take some measurements of the environment. Depending on whether you're using the physical HAT or the emulator, you will need to change the first line of the code.

If you are using the physical SenseHAT on a Raspberry Pi or trinket, start the code with:

~~~~
```python
from sense_hat import
SenseHat
```
~~~~

If you are using the SenseHAT emulator you will need to start the code with:

~~~~
```python
from sense_emu import
SenseHat
``` (HW)
~~~~

ACTIVITY 1: HOW HOT IS IT IN THE JUNGLE? 25 MINUTES

We want to find out the temperature of the jungle so we can fetch the current temperature value and print it out. Here's the code we need to start:

~~~~
```python
sense=SenseHat()
temp = sense.temp
print(temp)
```
~~~~

What do you notice about the temperature?

We need to change the temperature and make it more readable by removing the extra digits after the decimal point. Again, here's the code to get things started:

~~~~
```python
temp = round (temp,2))
print (temp)
```
~~~~

Now we have an accurate measurement of the temperature in the jungle, we need to know if it going to be safe for us to survive.

If your body temperature gets higher than 38.3 degrees C, you will enter hyperthermia and if it drops below 10 degrees C, you run the risk of hypothermia:

~~~~
```python
if temp < 10:
    print (“Warning too cold -
risk of hypothermia”)
elif temp >38:
    print (“Warning too hot -
risk of hyperthermia”)
else:
    print (“Safe temperature”)
```
~~~~

Extending the task

Measuring the temperature once is useful, but why don't we read it every five minutes?

Task

Create a loop to repeat the temperature measurement.

Add a delay of 300 seconds.

ACTIVITY 2: IS IT GOING TO RAIN? 25 MINUTES

Weather forecasters use air pressure as a way of predicting the weather. You might have seen a barometer before or even have one at home. In this task, we are going to use the Raspberry Pi to log the air pressure and predict if a storm is coming.

Start a new Python3 file and this time we will be collecting the air pressure:

~~~
```python
from sense_emu import SenseHat
sense = SenseHat()
while True:
    pressure = sense.pressure
    print (pressure)
```
~~~

Run the code and watch the long list of air pressures disappear in front of you – this is not going to be the simplest way to observe a change in air pressure. The SenseHAT can measure a pressure range from around 250 mbar to 1250 mbar. We can use the pixels on the SenseHAT to show the pressure as a bar chart.

~~~
```python
from sense_hat import SenseHat
from time import sleep
sense = SenseHat()
r=[255,0,0]
sense.clear()
while True:
    for c in range (0,7):
        pressure = sense.
pressure
        print (pressure)
        graph_pressure =
int(pressure / 150)
        print (graph_pressure)
        for i in range(graph_
pressure):
            sense.set_
pixel(c,i,r)
        print (c)
        sleep(300)
    sense.clear()
```
~~~

The actual value of the pressure is divided by 150, to give a value between 0 and 8 which can be displayed on the SenseHAT pixels as a bar. Every ten minutes a new value is taken, and the next bar is displayed. After all eight bars are displayed, the loop restarts. Over the hour you can monitor the weather by looking at the air pressure.

If the air pressure is high we can predict clear skies, light winds, and bright weather. If the air pressure is low we can predict we are in for cloudy and wet weather. If the air pressure starts to drop we can predict a change in the weather and we might need to get out our raincoat or umbrella.

SPENCER ORGAN

Spencer is a chemistry, physics and Computer Science teacher, Raspberry Pi enthusiast, and Certified RPi educator from the West Midlands. He loves running workshops and building Pi projects.

> “ OVER AN HOUR YOU CAN MONITOR THE WEATHER BY LOOKING AT THE AIR PRESSURE

LESSON PLAN INFORMATION FOR TEACHERS

These activities can be used in both computer science lessons or in a cross-curricular science project. The activities will give students opportunities to take real world measurements with the SenseHat.

Activity 1 introduces students to taking measurements from the SenseHat and formatting the results to make them more readable. The activity also gives students an opportunity to use if... then... statements to respond to the measurements.

Activity 2 builds on the first task, and takes measurements of atmospheric pressure and displays them in a useful format as a bar chart on the SenseHat LED matrix.

ASSESSMENT

- How would you change the colour of the bars to show a drop in air pressure?
- Could you trigger an alert if the pressure drops dramatically in a short space of time?

AGE RANGE

7-10

LESSON TYPE

Physical computing

REQUIREMENTS

- Pro-Bot
- Large sheets of paper
- Felt-tip pens

SHAPE UP WITH PRO-BOT

Students enrich their maths knowledge through tackling challenges and programming a robot

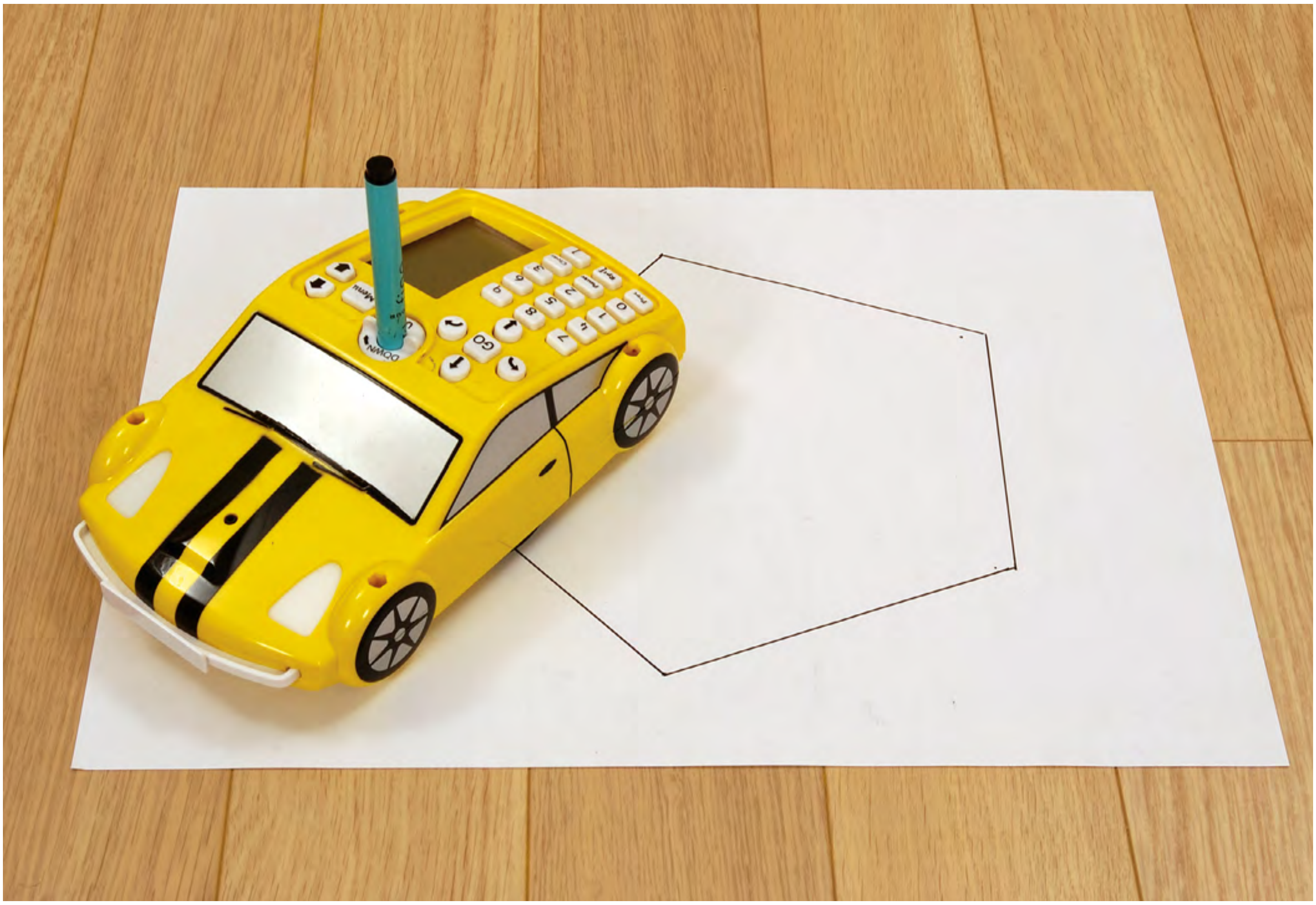
Pro-Bot is a floor robot that is programmed using a Logo-like programming language. The robot is able to carry a pen that allows it to draw lines on large sheets of paper.

In this lesson, students enrich their maths knowledge by working on real world challenges. Students begin with some collaborative, unplugged work. This then leads them into planning and writing programs on Pro-Bot. The lesson assumes that students have had some previous experience of Pro-Bot. Students will come to this challenge knowing how to enter and execute programs that make Pro-Bot move forward and turn.

This plan follows on from a maths lesson where students will have been working on area and perimeter. (HW)

OBJECTIVES

- ✓ Explore maths concepts of perimeter and area using Pro-Bot
- ✓ To apply maths skills to real-world challenges
- ✓ To write programs which accurately represent shape



ACTIVITY 1: PERIMETER CHALLENGE 60 MINUTES

Understanding the problem

The teacher shares the challenge with the class: A farmer needs to build an animal enclosure. The farmer has only 36 metres of fencing available.

Working unplugged

Students work in groups to investigate how many different ways the enclosure can be built. Initially they use pen and paper to sketch out their ideas. Students are told to use 1 cm to represent 1 m in their workings.

Programming task

Once the students have come up with at least three options, they will need to program Pro-Bot to draw each option.

Plan, do, review

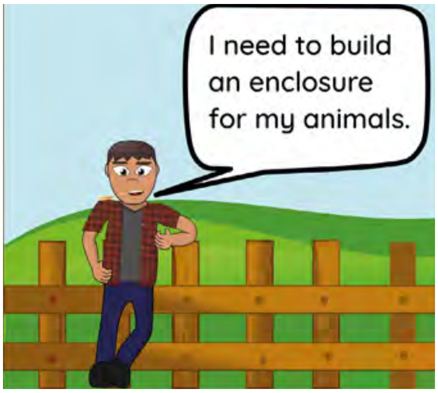
Students are encouraged to look carefully at their polygon animal enclosure sketches. They are asked to consider angles of turn and length of sides. Students discuss this information and use it to plan the program for their first enclosure. The students are then ready to program Pro-Bot and review whether their first program accurately represents their first sketch. Their review process should include measuring of sides and angles. Any discrepancies between their sketch, plan, and Pro-Bot output will need addressing.

Sometimes students will need to review their maths and sometimes they will need to debug their program. This 'plan, do, review' process is then repeated for the second and third animal enclosures.

When all enclosures are successfully drawn, students label each with the appropriate polygon name.

Extension challenge

The teacher shares a new problem with the class. The farmer has miscalculated, and actually only has 35 metres of fencing available. The farmer wants a square or rectangular fence using all of the fencing material. Students are asked to consider whether this is possible and to explain why or why not.



ASSESSMENT

How will you know whether your students have met the learning objectives? You could include some questions to ask here, or describe other methods you will use to assess progress.

DIFFERENTIATION

Both challenges can be made simpler by using smaller numbers and made more complex by using bigger numbers. Children needing additional support with the unplugged activity could be given concrete resources.

DI NORVOCK

Di is a Year 6 teacher in a Primary School in Sheffield. She has been a computing subject leader and been responsible for whole school curriculum development.

ACTIVITY 2: AREA CHALLENGE 60 MINUTES

Understanding the problem

A farmer wants to put solar panels in one of their fields. The farmer can buy 100 m^2 of panels. The panels need to be placed together to make a square or rectangle.

Working unplugged

Students work in groups to investigate the different ways the solar panels can be arranged. Initially they use pen and paper to sketch out their ideas. Students are told to use 1 cm to represent 1 m in their workings.

Programming task

Once the students have come up with at least three options, they will need to program Pro-Bot to draw each option.

Plan, do, review

Once again, students should plan their ideas before programming the Pro-Bot. The difference this time is this is an area, rather than a perimeter, challenge. As such, their review process should include measuring the sides and calculating the area. Any discrepancies between their sketch, plan, and Pro-Bot output will need addressing.

Sometimes students will need to review their maths and sometimes they will need to debug their program. This 'plan, do, review' process is then repeated for the second and third solar panel plans.

FEATURE

Code Clubs for teachers can help build confidence ahead of working with technology in the classroom

TEACHERS NEED CODE CLUBS TOO

Dr. Ursula Martin suggests that teachers could benefit from participating in a Code Club to help conquer their fears of integrating technology into their classrooms

Many teachers fear using technology in the classroom, because they worry they would be unable to control what the students have access to on their devices. Their fears of technology integration become a disadvantage to their students, who need and want the exposure to technology use. The world they're entering, after all, is technology-dominated.

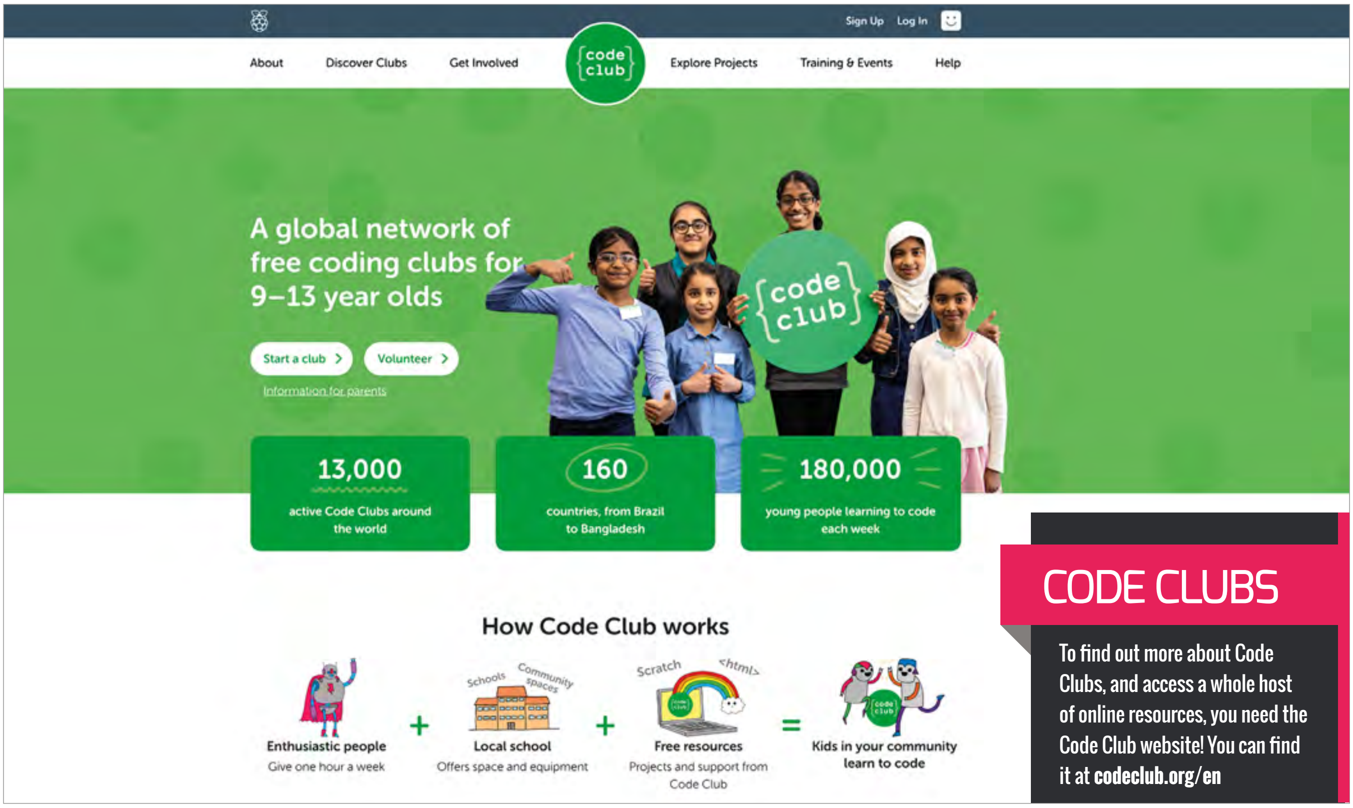
When we prepare teachers to 'conquer their fears' of integrating technology into the classroom, we prepare them to help students reach their potential. Introducing students to different types of technology apps, programs, and tools in the classroom helps to prepare them for their tech-driven future. Allowing students to use technology in the classroom opens doors to many opportunities for students to collaborate, use their critical thinking skills to solve problems, communicate with each other as well as with their teachers, and lastly – but most important – create.

Creating using code

The integration of coding in the classroom has offered one of many opportunities for students to create using technology. Coding is as relevant a skill, for jobs of the future, as the content currently being taught to students in the classroom.

Coding in many school districts in the US begins in kindergarten, and usage continues through high school. That's because school districts have recognised that allowing students to be creative activates their critical thinking skills, and thus more and more districts are introducing coding into their schools.

Many of these schools have found ways to integrate coding into their curriculum but for those that have not, Code Clubs have been the alternative. Whether coding is integrated into the classroom, or is offered in an after-



Sign Up Log In

About Discover Clubs Get Involved {code club} Explore Projects Training & Events Help

A global network of free coding clubs for 9–13 year olds

Start a club > Volunteer >

Information for parents

13,000 active Code Clubs around the world

160 countries, from Brazil to Bangladesh

180,000 young people learning to code each week

How Code Club works

Enthusiastic people Give one hour a week + Local school Offers space and equipment + Free resources Projects and support from Code Club = Kids in your community learn to code

CODE CLUBS

To find out more about Code Clubs, and access a whole host of online resources, you need the Code Club website! You can find it at **codeclub.org/en**

> MAKE SURE TOO THAT TEACHERS HAVE ACCESS TO THE TOOLS THAT WILL BE USED

school club, teachers must be prepared to help students understand how to write code, and what tools to use to activate the coding.

Code Club for teachers

Having a Code Club for students to teach them how to code, and use tools such as Raspberry Pi units to run that code, is a wonderful idea. But how do we get teachers prepared to teach students these skills?

We start a Code Club for teachers. Teachers need Code Clubs too, to help them better understand how coding works, to allow them to better teach their students. The best way to prepare teachers to teach their students how to code is to have models of what a Code Club looks like and how it works. This offers teachers the opportunity to participate in a Code Club too. They have the opportunity to experience coding the way their students would, and it also helps them to conquer their fears of integrating technology into their classrooms.

Getting started

Start by scheduling a time that teachers can meet for the first Code Club meeting. This may be after school or during the day, with substitutes provided (if possible). Use this meeting to discuss what times would be best for future meetings, and schedule those dates. Also, introduce teachers to what coding is and a few tools that will be used in future club meetings such as Raspberry Pi, Dash and Dot, Tynker, Drones, and such like.

Make sure too that teachers have access to the tools that will be used in the Code Club meetings. These may be the tools the teachers already have in their schools that are not currently being used. If teachers do not have access to coding tools, decide how these tools will be acquired.

Once tools have been acquired, each future meeting should be spent modelling what Code Club activities will take place once teachers have started their own clubs with their students. This should consist of learning coding languages, along with demonstrating and sharing with each other what was learned in the meeting.

Code Club for teachers should exist for at least a semester before teachers branch out on their own to start a club of their own. The suggested time for each club meeting is no longer than two hours, so as to not overwhelm the teacher.

Also, this will give the teachers time to take in what they have learned, and to be able to practice those skills during the time between meetings. This should then ensure that the teacher is comfortable with how to implement what they've learned with their students. Ultimately, a successful Code Club for teachers should, in turn, ensure a successful Code Club or class for their students too. (HW)

DR. URSULA MARTIN

Dr. Ursula Martin has a background teaching high school biology, anatomy and physiology, and environmental science. She is currently a district level Instructional Technology Resource Teacher. She has been an educator for 17 years.

MAKE COMPUTING SPARKLE FOR OPEN EVENING

In this guide, **Alan O'Donohoe** of exa.foundation suggests how teachers might go about presenting their Computing curriculum area in the best light during a school open event

Whichever educational setting you work in, it is likely that you will have encountered the concept of an open evening or open day, and may have already experienced the frenzy of activity that leads up to the event. Although these events tend to be more common in the secondary and higher education sectors, there are signs that they are becoming more popular in primary and nursery education too.

Traditionally, schools and colleges have hosted open evenings to showcase the facilities to their community, with the principle aims being to:

- Market the school or college in general while promoting its facilities, values and culture
- Present the greatest strengths of the school or college in the most favourable light
- Persuade prospective students and parents in the process of selecting a school or college, that this is the school they should choose
- Allow visitors opportunities to learn more about curriculum provision and meet the staff

It can be a very rewarding experience to have interested guests visit you in your classroom because they want to discover more about the subject you teach, to look at the facilities and resources available, and admire your classroom wall displays (see 'Insiders Guide' in *Hello World* Issue 9). There is no doubt that preparing for an influx

Showing off fun activities

of visitors to your classroom can lead to increased workload. The intention of this guide is to minimise the stress, suggest time saving tips, and lead to better outcomes.

On those occasions when we find ourselves under the pressure of an impending deadline or important occasion, we don't always perform at our best. This is particularly true during those times when we experience a heavy workload, so exercise caution around some of the common mistakes that a busy teacher should avoid falling into. These, for instance.

Leaving it too late to recruit volunteers

Having enough of the best calibre of student volunteers to support your activities can really make a massive difference to the success or failure of your event. Though this seems so obvious it doesn't need stating, on too many occasions my workload has been so heavy in the lead-up to open night, that I've overlooked volunteer recruitment until it's too late. By that stage, I have ended up desperately trying to recruit any student who is available on the night, since my ideal candidates have already agreed to support activities in other areas around school.

Showing off your shiny kit

It's become a no-brainer in some quarters to rely on simply having enough attractive, new flashy pieces of educational technology, hardware, or software on display during open evening to woo visitors' attention – and then hope that this alone will be enough to

Hopefully satisfied customers!

ALAN O'DONOHOE

Alan (**@teknoteacher**) has more than 20 years' experience teaching and leading Technology, ICT, and Computing in schools in England. He runs exa.foundation, delivering professional development to engage digital makers, support the teaching of computing, and promote the appropriate use of technology.

stimulate the interest of visitors. It's certainly a helpful strategy for the busy teacher, since there will no doubt be high levels of interest around the shiny kit, especially if this is unfamiliar to your visitors. The problem is that this does not necessarily convey a typical experience to visitors, particularly in terms of learning experiences.

If you do choose this route, consider having a series of questions pre-prepared to prompt discussion with visitors:

- What positive effects do you think this technology will have on us?
- What potential do you think this offers in a learning context?
- Where do you think there may be potential for harm or abuse of this technology?

Too much is worse than none at all

Another common issue is having too many different activities on offer in your Computing classroom, which may just overwhelm your visitors altogether.

When I asked other Computing teachers to tell me how they typically organise their open event, one suggested ten activities that they usually have on offer! As it is, an open evening can be a very busy event across the whole of the school with many different activities all vying for your visitors' attention. Before they arrive in your classroom, if your visitors have previously visited seven other classrooms that evening, they may already be suffering from stimulation fatigue. By opting for the 'less is more' approach instead, having no more than three well-planned activities for visitors in your teaching space, that would have the effect of allowing visitors adequate time to explore these activities properly.

Being too busy to talk

If you have planned a comprehensive programme of activities to take place in your classroom, this is likely to demand much of your attention and require high levels of supervision from you, which will inevitably limit the amount and quality of time you have available to have meaningful discussions and interactions with visitors.

Instead, you should plan for your classroom to be a calm environment where there are opportunities for you to learn as much about your visitors and their interests as they might learn about you and your subject. Inevitably, visitors engaging in quality discussions are more likely to walk away with a positive impression of Computing in your school, and in the process you will have gained a deeper understanding of their impressions.

The one-person band

In our school, for many years we had a long-established habit of having activities on offer across all three of our Computing suites on open night, and in each suite, one of our Computing teachers supervised the activities on offer. At the end of the event, each of us was totally exhausted from the effort required to sustain visitors over a three-hour period. Then one year we plotted a different approach: we would join forces so that all of us would be together in the same room all evening.

This was a far better solution in many ways: by combining our efforts together in one classroom instead of three, we were able to ensure that we were better resourced for welcoming visitors, and able to provide a better experience for visitors too.

Too shy to try

While many visitors will quite happily engage with the activities on offer and participate in discussions during open night, some visitors may be so shy or polite that they need gentle prompting or persuasion to engage in the activities on offer. Taking care not to be too pushy or overly assertive, I recommend your young volunteers practice inviting the quietest of visitors to take a closer look, and try some of the activities. The question, "would you like to try this

> ANOTHER COMMON ISSUE IS HAVING TOO MANY DIFFERENT ACTIVITIES ON OFFER IN YOUR COMPUTING CLASSROOM

It's the children who'll sell the subject best

You can't beat a BBC Micro!

activity here?" is open-ended enough to allow your most timid of visitors to politely nod and take a step backwards if it's not their thing. See the advice on using Computing Ambassadors.

Computing Ambassadors

Recruit volunteers from early on, then train them to engage with visitors. Suggest a three-point strategy for them: Welcome, Offer, Support. A little time spent training your Computing Ambassadors will pay off during the event.

- Welcome – This can range from a friendly welcoming smile to a "Hello! My name is Alan. May I ask your name?"
- Offer – Suggest some of the activities available, "Would you like to try debugging some GCSE Computer Science programming projects?"
- Support – Stay with them and lead them through the activity they choose, offering praise or support as appropriate

I spoke to educational technology Ellie Overland – who oversees secondary ITT in computing at Manchester Metropolitan University – about computer ambassadors, and she said that, "I have recently been to visit some fabulous open evenings. It was the children who really sold the subject, they were leading the activities and talking passionately about the subject".

Activities need to be memorable for your visitors...

Suggested examples of activities

If you plan to have three different types of activity on offer as suggested below, it means you will be able to offer your visitors different types of experience depending on their motives for attending the event

1. Pupil work display

Aim: To present visitors with a snapshot view of the learning that takes place
Audience suitability: visitors of all ages

Have some examples of pupil work on display to show some evidence of typical learning experiences in the classroom. In a primary setting, these could include games and projects that children have created using Scratch, with opportunities for visitors to try out the projects and offer some feedback to contribute to the development.

In a secondary setting, you might include some previous solutions to GCSE programming projects to demonstrate the level that students have been working on. If you had some 'sabotaged' examples of the students' coded solutions, you could ask visitors if they could spot the syntax errors or semantic errors that had been deliberately added in, to test their observational powers, leading into a discussion about debugging. Alternatively, you could have some sticky labels for visitors to attach to relevant sections, for example sequence, selection and iteration.

2. Hands-on experiences

Aim: To provide visitors with enjoyable and memorable experiences
Audience suitability: younger visitors

If you choose to have some hands-on activities for visitors to try themselves, make sure that these are accessible to those with

. ..and fun too!

little prior knowledge or experience. You could have a range of challenges that gradually increase in difficulty. Make sure that your computing ambassadors know to keep their hands away from the keyboards and mice when the visitors are trying computer-based challenges. I have found that nested challenges based around the use of Minecraft Pi or Mozilla X-Ray Goggles have the right balance of fun, engagement, and challenge. However, they do take some time to set up. Some teachers have enjoyed a lot of success with an activity that produces a physical outcome that visitors can take away as a keepsake.

3. Talking points

Aim: To stimulate some discussion with adult visitors
Audience suitability: adult visitors

You could demonstrate some artefacts that make use of new and developing technologies, or dust off your collection of artefacts from the history of computing. You will find that objects like this will help stimulate all sorts of interesting discussions. Whether you choose to show off the latest augmented or virtual technologies or a collection of floppy disks, vintage games consoles and controllers, some adult visitors will find their curiosity gets the better of them and will want to ask questions to discover more. That, or they'll want to take you on a trip down memory lane with them while they regale you of tales of a misspent youth playing video games, or how they built an Asteroid clone in 6502 assembly language.

Suggestions from teachers

1. Pupil work display

- **Tom Rattle:** Since our use of Scratch is now predominantly cloud-based when we use it with our classes, it's really easy to show off some of the most impressive pieces of Year 7 work to give a taster of what they'd be doing in their first year.
- **Paul Powell:** I organise work from all year groups on display, and for programming in particular, I show progress from Year 7 through to Year 11. I find this really helps to engage those parents with an interest in the technical aspects.

Have pupils on hand to help

Be on hand to give guidance where needed

- **William Lau:** I've had some GCSE students in to work on their GCSE programming project. It didn't count towards their 20 hours and they knew it wasn't graded. However, they simply wanted to get their head around 2D arrays and file writing and they figured that three hours one evening in school would be better than sat at home. Eight months later, the same students who took part were all on track and predicted 8s and 9s or A*s respectively.

2. Hands-on experiences

- **Matt Moore and Jan Dowding:** Kodu. Makey Makey with Play-Doh or fruit. Minecraft server with large-scale build challenges.
- **Penny Cater:** Crack The Code – have a safe filled with sweets and visitors have to solve a binary puzzle to find the number to unlock it.
- **Jamie Edmondson :** Parsons Problems using Scratch, putting the blocks in sequence to tell a simple joke. Use a GreenScreen app like Do Ink to place visitors in the places they would most like to visit. With a mobile device connected to a large screen, visitors could use a coding app like A.L.E.X or Lightbot to solve the different challenges.
- **Katie Vanderpere-Brown:** I have shared resources for 'Binary Muffins' activity on the Computing At School community site here – **bit.ly/2mcZS58** [login required], and 'Journey of a selfie', here – **bit.ly/2ILKH2w**.
- **Claire Buckler:** Our most popular idea has been a broken apart Pac-Man game, built using Scratch which visitors had to solve

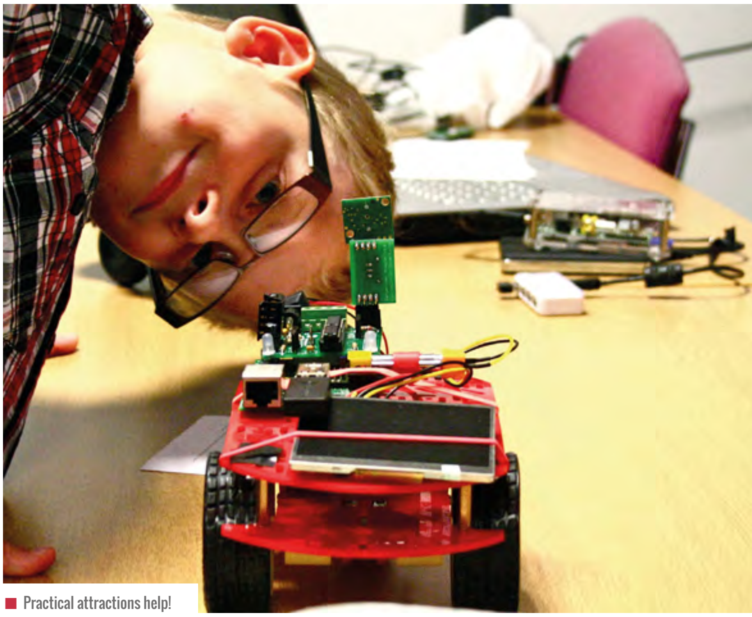
> CRACK THE CODE – HAVE A SAFE FILLED WITH SWEETS AND VISITORS HAVE TO SOLVE A BINARY PUZZLE TO FIND THE NUMBER TO UNLOCK IT

together. Other problem-solving puzzles like peg puzzle or Towers of Hanoi have been popular.

- **Sarah Twigg:** A nice idea I've tried is to have a message written in binary on the board, for instance 'Welcome to Computing', and provide visitors with the ASCII table, to help them to convert it. If they are successful then they receive a little certificate or reward.
- **Michael added:** A variation on Sarah Twigg's suggestion, we had the ASCII board up and binary equivalent, and made little cards so they could write their names or initials in binary.
- **Amanda:** Visitors made key chains with their initials encoded in ASCII that they could take away. We bought lots of empty key chains and a bag of two-colour pony beads, and visitors followed a guide as to which colour went where. We used wool, but a stronger cord would work better, threaded it through, and left them so just a link with the wool was attached and the visitors could do one initial or two.
- **Chris Sharples:** A Digital Leader wearing a sorting hat (sourced off eBay) with a micro:bit supposedly sorting into Hogwarts houses (but really Griffindor!) and a dance mat for squashing bugs using Raspberry Pi (code courtesy of University of York Computing Department).

3. Talking points

- **Penny Cater:** We have a 3D-printed Rubik's cube solver, Lego Mindstorms, and Mambo Drones that I get students to prepare the week before and run the show.
- **Alex Clewett:** Set up a game like **@KeepTalkingGame** using Oculus on Gear VR or another VR headset.
- **Oli Howson:** Get the Sinclair Spectrum out! (Other 8-bit computers also available).
- **Tom Rattle:** I brought a Sphero BB-8 in from home and recruited some Year 9 students to show visitors how to code the blocks to make it roll around. That drew a small crowd and didn't take much effort beyond pairing it to the iPad with Bluetooth.
- **Donna Shah, Alicja Wojtowicz, Ben Barnes, and Jan Dowding** all recommended children demonstrate some micro:bit projects that they have built and have projects available that visitors can try building themselves.
- **Paul Powell:** I connect RetroPie (a Raspberry Pi-based games emulator) to the big screen which keeps the room busy, and also prompts conversations about the Raspberry Pi computer. (HW)


Practical attractions help!

FURTHER INFORMATION

For more free, friendly advice about planning the best kind of open evening experiences, contact the author of this guide Alan O'Donohoe via **alan@exa.foundation** or **@exafoundation** on Twitter.

Getting hands-on

8-bit classics!

YOUR QUESTIONS

If you have a question you'd like the Learning Team at the Raspberry Pi Foundation to answer, contact us on Twitter via **@HelloWorld_Edu**. Alternatively, email us with 'Question' in the subject line (**contact@helloworld.cc**).

Q HOW DO YOU TEACH CODING TO POST-PRIMARY (11 - 16 YEAR-OLD) CHILDREN AND KEEP THEM ENGAGED?

- CIPRIAN MANEA (@OVIPIC), VIA TWITTER

A

CARRIE ANNE: Finding interesting 'real world' contexts can be a great way to keep learners engaged with programming beyond primary. Like taking part in a robotics competition or having your code run in space by astronauts on the International Space Station! The next European Astro Pi Challenge launches 18 September and comprises two missions: Mission Zero, the simpler of the two, and Mission Space Labs, which gives you the chance to have your scientific experiment run on the ISS. Visit **astro-pi.org** to get started and check which countries are eligible to have their code run in space.

ALLEN: I try to include activities that link things children are already familiar with to concepts I'd like them to learn. In doing this, I find it useful to build in lots of time to modify and experiment with concepts taught along the way, so it's not too prescripted. When there is too much direct teaching of programming for long periods, learners can become disengaged. Building on the simple idea of a chatbot for instance, learners always like to come up with fun questions and responses; once you have the buy-in from learners and they are enjoying the journey, you can then thread in more complicated concepts or data structures to improve their programs. When learners have ownership over their own code, as opposed to being asked to copy code, I find they are more willing to push themselves, and those learners have then independently gone on to discover how they can do something themselves, rather than wait for it to be taught.

Q I'M STARTING TO PLAN NEXT TERM'S SCHEME OF WORK FOR YEAR 6. I WANT TO DO A TEXT-BASED PROGRAMMING LANGUAGE. WHICH WOULD BE BETTER: PYTHON OR JAVA?

- JAMIE BOOTH, VIA FACEBOOK

A

DUNCAN: Out of interest, why look to do a text-based language at Year 6? There are many computing concepts that can still be covered in Scratch. Have you explored 'Make your own blocks' and passed values into them? This is a key concept, helps with decomposition, generalisation, and cognitive overload. What about list? Also, have you looked at broadcast messages between sprites? Computationally, there's much that can be done in Scratch.

LAURA: I'd highly recommend Python over Java for primary pupils because it doesn't need to be compiled, so the installation process to get it working is less complex, and writing a simple program requires a lot less 'boilerplate' set-up code. It's possible that you may actually be thinking of JavaScript, not Java – confusingly, they're two separate languages. JavaScript would be a reasonable language to introduce to Year 6 pupils because it requires minimal set-up (you can write JavaScript programs with a web browser and a text editor) and it's a very popular language at the moment, used on a huge amount of websites.

You can find Python and JavaScript resources at **rpf.io/learn**.

Q DURING STAFF SCRATCH TRAINING, WHAT WOULD BE A GOOD APPROACH TO PRACTICE SOME BASIC SEQUENCE AND REPETITION SKILLS WHILST EXPOSING ATTENDEES TO THE SCRATCH PLATFORM? - LOUISE O'MAHONY, VIA FACEBOOK

A

GEORGE: There are quite a few relevant projects available in the Scratch tutorials. Projects like 'Animate a name' and 'Make it fly' cover the essentials and are quite generic. The participants are also likely to check out and follow other tutorials after training, to practice or explore different features. Another excellent place to start are the code club projects in the first Scratch module.

ALLEN: My favourite go-to when introducing Scratch is the modelling of a set of traffic lights, using simple shapes as sprites to represent the individual lights. Everyone knows the sequence of traffic lights, and using this familiar real-world scenario, attendees can translate what they know into actions in Scratch.

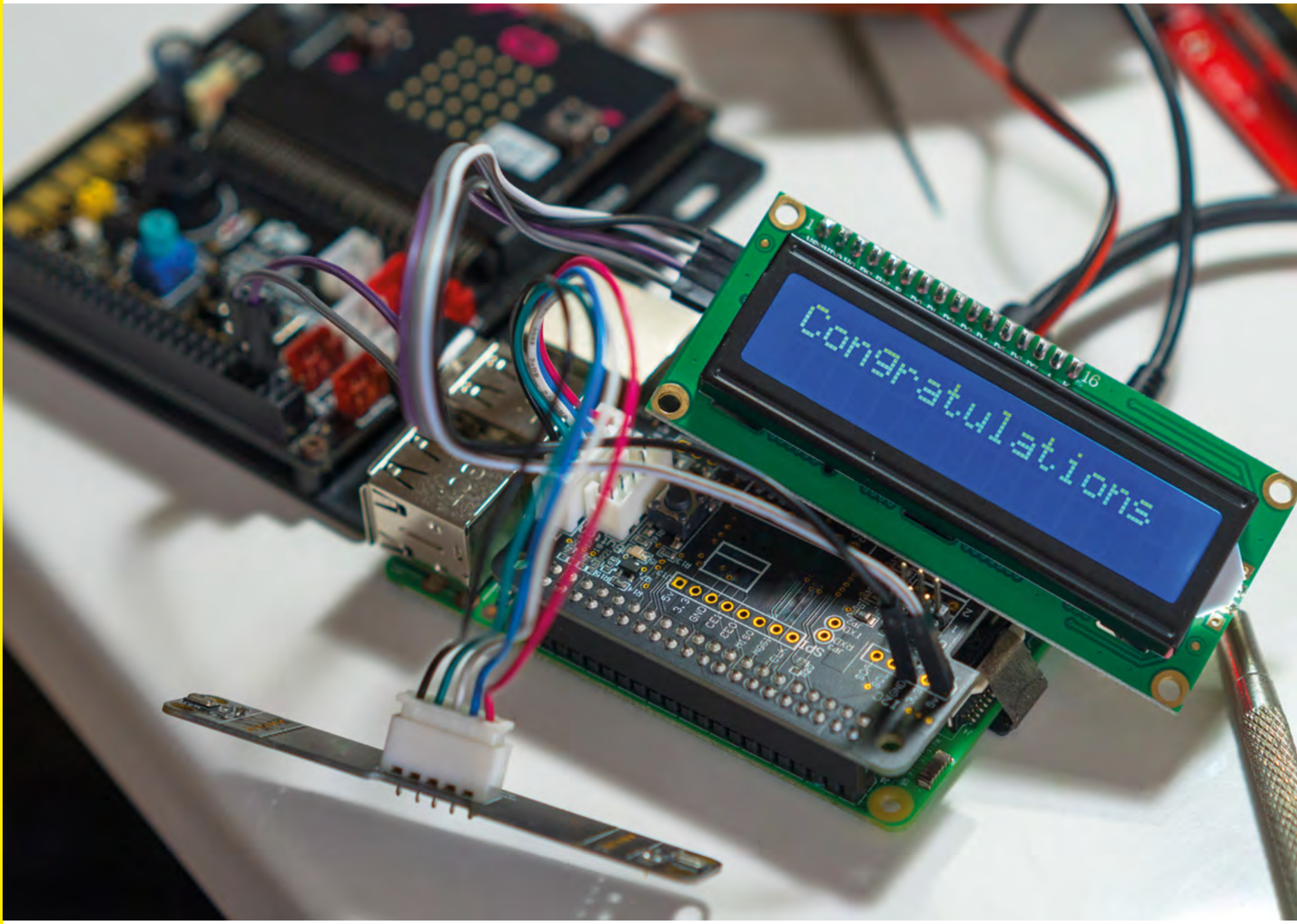
In doing this, they are utilising sequence and repetition in an activity that visually – and immediately – allows them to see if they have been successful, and if not, they can visibly see where they have gone wrong. Moving forward, they then take appropriate steps to modify their program to make it function as it should.

Q WHAT IS YOUR 'MUST-HAVE' EDUCATIONAL TECHNOLOGY IN THE CLASSROOM, AND WHY? - CAROL STUART (@CASCHAT), VIA TWITTER

A **SWAY:** When choosing technology to include in the classroom, it's important to consider why it is being introduced and how it is adding to the learning that is already taking place there. The SAMR model may help you decide whether a specific piece of technology is right for the task you're doing.

This model allows you to consider whether the technology you're choosing is 'substituting' a paper-based method, 'augmenting' it (adding a little more functionality, but not huge change), 'modifying' the activity in a way that could not be done without the use of technology, or finally, 'redefinition', which involves completely transforming the task's design. Technology that is transforming learning opportunities in your classroom is a better investment that something that only enhances it.

ALLEN: I don't have a 'must-have' technology in the classroom, as teaching should be driven by what you want the children to learn. In some cases such as physical computing, my must-have would be Raspberry Pis or Crumbles, whilst if I were delivering data representation I would more often than not opt for unplugged activities. I think there is a real danger of recommending must-have tech as it can wrongly be seen as a magical panacea that will solve your problems, and in truth, it won't. Look at the learning you want to take place and select the best (or one of the best) tools for the job. Any educational technology is only as good as the lesson and the planning that goes with it. How many cupboards in schools are full of 'must-have' tech that has been purchased following poor advice or lack of training/understanding?



Q DO YOU TEACH A DIFFERENT PROGRAMMING LANGUAGE AT A-LEVEL TO KS3/4? - @CASCHAT, VIA TWITTER

A **ALLEN:** Personally, no. Given that once students progress to A-level they usually have to produce a complex solution to a problem of their own choosing, I find it better to build on the knowledge they already have. In employing this method, I find that learners are less afraid of undertaking such a project as they have built up their confidence using the language (Python in my case) for a few years. They now understand the fundamental building blocks and concepts, so they find it easier to build on this without having to learn the syntax of a new language at what is often a critical time for students. That is not to say I haven't had students experiment with other languages during their course, it's just not so mission-critical where they need to learn a lot in a short space of time.

Q WHAT KINDS OF FORMATIVE ASSESSMENTS HAVE YOU FOUND HELPFUL IN ASSESSING STUDENTS' UNDERSTANDING? - @CASCHAT, VIA TWITTER

A **GEORGE:** For concept-rich units, my personal favourite is using short quizzes in Google Forms. I will let students take the quiz and then view the graphical summary of the answers on the board, together with them. This allows me to provide immediate whole-class feedback and instantly identify problems and misconceptions: questions with a high percentage of wrong answers or questions with an even distribution across all answers indicate where I need to focus and provide clarifications.

BEN: Kahoot is also very effective for multiple-choice questions or polls.

ALLEN: I have used fairly large programs that include lots of errors, some logical, some syntactical, and some run-time. In getting learners to 'fix' the code so it runs properly, you get to understand where learners have understood, have the capability to correct code, and can also see any common areas requiring improvement across the class. When doing this, I ask learners to comment specifically on what they have changed and why, so you can gauge the depth of their understanding. You can really probe what a learner can do by asking them to either complete or fix code. There is little chance learners can guess the right answer, plus it's great for informing your ongoing planning.

Q HOW DO YOU GET YOUR STUDENTS TO EVIDENCE THEIR LEARNING AT KS2? HOW OFTEN DO THEY DO THEY DO THIS? - MISS C (@TECH_MISSC), VIA TWITTER

A **BEN:** This is a very open-ended question and the answer depends on a number of factors – your school's expectations, the equipment you have in school, and your own preferences as a teacher. For offline or unplugged tasks, a computing folder or journal can be useful. The frequency of work will vary according to the learning taking place; however, even for computer-based tasks or units, it's useful for learners to plan or reflect on paper. Most popular computing resources such as Scratch offer functions to enable students to save their work online and in structured folders. These can be evidenced through links or QR codes.

Another popular example is to use a school/class blog (either public or private) to showcase their achievements. This will also act as evidence of what you have covered and the quality of work the students have produced. Linked to this, the children could create online portfolios where they choose to add work they are proud of at the end of a lesson – this could be photos of paper-based activities as well as links to digital content. There are many platforms that offer facilities like this, but even individual documents or presentations which the students can access could be used.

YOUR LETTERS

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**. Alternatively, email us with 'Teacher Letter' in the subject line (**contact@helloworld.cc**).

Inclusion

Dear Hello World,

I was very pleased to read the letter from Ilija Jovanovic in issue 9 of Hello World, about the wish for more focus on including learners with disabilities in computing. I found myself nodding furiously at the letter, and am gratified that you will be devoting a future issue to this topic.

Ilija is absolutely right in my view: this is a national crisis that's getting left behind by successive governments who – let's charitably suggest – have different priorities. We have a generation of youngsters who have shown they want to embrace and get involved in computing, but disability is proving to be a barrier. I appreciate this is a generalisation, but in my view, it shouldn't even be a sniff of a problem. Technology has been such an enabler, such a powerful force for many who live day to day with accessibility challenges and/or disability. There's no lack of will or intent on the side of the pupils. The barriers are being put in by budget cuts, targets, and the lack of a cogent national strategy.

I hugely commend and applaud the many educators around the country and around the world who have rolled their sleeves up and done something about this, who have seized the initiative, introducing ideas and innovations within the schools and colleges where they work. Without these people, children who deserve a whole lot better, as Ilija observes, would get left behind.

J N Anderson

Thank you for your letter.

As we wrote in our last issue, this is a subject very high on our own agenda, and the next edition of Hello World – due out in November – will have a specific theme of diversity and inclusion. We encourage anyone who wishes to add to that issue to get in touch: our email address is **contact@helloworld.cc**. We'd love to hear from you.

Steampunk

Dear Hello World,

Aisha Flores' article in the most recent edition of your magazine (issue 9) was a real delight to read. Much of the conversation in computer science teaching of late has been centring on attracting more and more girls to the subject, particularly at GCSE level, and engaging them with it. The Buttercup STEAM Camps that Aisha described sound wonderful, and I hope they come to the UK soon.

Please keep covering such ideas in your magazine. We're seeing a continual uplift in the number of girls pursuing coding, computer science, and technology, and long may that continue.

Robin Moore

It's very much our pleasure to feature articles such as Aisha's, and thank you for taking the time to write in. We're always keen to hear of any innovations and ideas people have to engage youngsters of whatever gender in coding and computing, and do get in touch if you have an article idea or suggestions for the magazine.

Taking games to the classroom

Dear Hello World,

I was really interested to read the article from Tom Bromwich in the most recent issue of Hello World, where he talked about embracing video games in the classroom as a way to engage his students. It was a fascinating read, and what I found particularly interesting was that he wasn't using specific educational titles, but commercial releases. Who can resist a little bit of *The Legend Of Zelda*?

I've oftentimes found a degree of snobbery towards video games as an educational aid, outside of perhaps something like *Minecraft: Educational Edition*. I've never understood it. Pretty much every pupil in my class has been playing some variant on *Fortnite* or *Minecraft* or *Roblox*, and I think it's an excellent entry point for getting pupils engaged in a topic.

Of course, as Tom outlines, nobody is expecting it to be about kids sitting around just playing Nintendo for an hour. I commend him enormously for using games as the foundation for getting his pupils writing. Please pass on my thanks to him for writing his article, which has certainly inspired a few ideas that I intend to take into the classroom. What are the chances of having a Hello World *Mario Kart* competition at some point in the future?!

B Harvey, London

Thanks so much for your letter, and for your kind words about Tom Bromwich's article. Gaming comes up regularly in Hello World, and it'd be remiss to say that teachers as a rule keep video games out of the classroom, but we do agree that it's more the creation of games than the playing of them that understandably tends to be the focus.

Gaming has evolved enormously, of course, and titles are now available that deal with very real issues in life, not least health and mental health. As a Trojan horse to getting pupils to express themselves, and perhaps open up a little, more and more are seeing the possibilities. If anyone reading has examples from their school, good or bad, we'd be interested to hear from you.

Sadly, we've no plans for a Hello World *Mario Kart* tournament just yet, though – but that's not to say we're resistant to the idea, of course...!

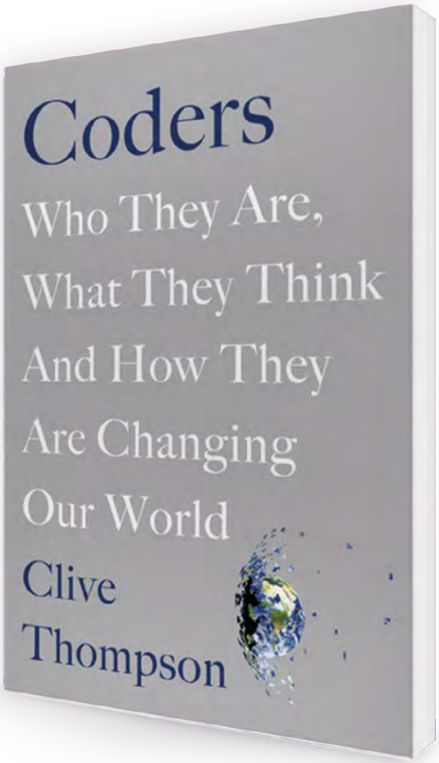
CODERS: WHO THEY ARE, WHAT THEY THINK AND HOW THEY ARE CHANGING OUR WORLD

INFO BY Clive Thompson | PUBLISHER Picador | PRICE £15.99 | ISBN 978-1529018981 | URL bit.ly/2SjcJNq

Thompson is a regular Wired columnist and an engaging, insightful author. This isn't a book about computer science, or even teaching CS, but it does an excellent job of giving the reader an idea of what coders do with their time. In Thompson's experience, much of this seems to be debugging their own and other people's code.

He does a great job of filling in the hinterland of the craft of software engineering for those who've not experienced it for themselves: the book is full of great anecdotes, many from the Silicon Valley big tech and start-up scene, which give the lay reader some feel for what the job of a programmer must be.

There's much here that will interest those who'd like to promote a more inclusive approach to computing education. Thompson deals well with the issues around the under-representation of women in CS, contrasting the early days with scores of female pioneers with today's all too masculine culture. He looks too at the high proportion of introverted coders, the belief in hyper-productive genius coders, the myth of meritocracy, and the 'move fast and break things' culture. The final chapter examines an initiative to retrain former coal miners as journeyman coders: CS for all!


Coders
Who They Are, What They Think And How They Are Changing Our World
Clive Thompson

Thompson is at his most engaging when describing his own attempts to walk-the-walk with coding, from his childhood BASIC programming through to wrestling with Twitter's API in Python. He confesses he finds coding more engrossing than writing, thanks to the challenge and feedback loop. He also deals with bigger issues well, including attempts to control cryptography and the current growth in machine learning applications.

Coders is a great read for CS teachers, both those wanting to get a better feel for what coding is like out in the real world and those wanting to make the subject that bit more inclusive in school. There are many sections here that would work really well as background reading for sixth form homework. I'd pop a couple of copies on the school library shelves, too: maybe one in the computing section, and another in careers. (HW)

CONCRETE MATHEMATICS: A FOUNDATION FOR COMPUTER SCIENCE

INFO **BY** Ronald L. Graham, Donald E. Knuth, Oren Patashnik | **PUBLISHER** Addison Wesley
PRICE £61.51 | **ISBN** 978-0201558029 | **URL** **https://bit.ly/2kIhfKv**

Knuth is, according to some, our greatest living computer scientist, with his *The Art of Computer Programming* the definitive, and still incomplete, text on algorithms. This book takes the foundational mathematics from there, expands on it, and takes a rather more discursive, accessible (but by no means easy) approach to these. For teachers, or students, who realise that many of the problems on Project Euler can be solved by just thinking about them rather than through coding, *Concrete Mathematics* teaches you how to do that thinking: "All you will need is a cool head, a large sheet of paper, and fairly decent handwriting."

Whilst the book first appears to be a disparate collection of mathematical techniques, there's an underlying message of how to solve problems by looking at them the right way: this is the computational thinking that has to come before the computational doing.

Given the co-author's invention of the typesetting language TeX and advocacy of literate programming, it should come as no surprise that this is a beautiful – and beautifully written – book. The notes in the margins are wonderful! (HW)

MACHINES LIKE ME

INFO **BY** Ian McEwan | **PUBLISHER** Jonathan Cape | **PRICE** £18.99 | **ISBN** 978-1787331662
URL **bit.ly/2FD7Mf3**

Booker prizewinner McEwan imagines an alternate history, where Britain lost the Falklands War, Benn became prime minister, and, more importantly for the plot, Alan Turing lives on to make human-level AI a reality with autonomous, humanoid robots available as the latest thing in consumer electronics.

The contrast between the dysfunctional narrator Charlie and the apparently perfect synthetic human (Adam) he buys is stark. Charlie is manipulative, expedient, and makes dubious moral choices; Adam is contemplative, cultured, and intuitive. The love triangle between Charlie, his neighbour Miranda, and Adam fuels most of the story, as well as scenes which might make teachers think twice before setting this as a class reader.

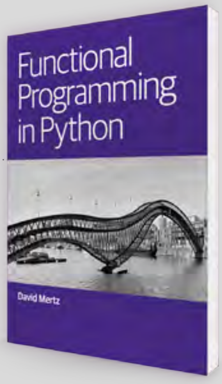
Turing's cameo role is nicely done, and McEwan does a good job of sketching out some key ideas in AI, and CS more broadly, but this is much more a book about being human, and attempting to be good, than it is about robotics or AI. (HW)

ESSENTIAL READING

Three books to get you talking about functional programming...

FUNCTIONAL PROGRAMMING IN PYTHON

BY David Mertz
PUBLISHER O'Reilly Media Inc.
PRICE Free online
ISBN 9781492048633
URL **bit.ly/2krRYEx**



Python isn't a functional language, but is popular in secondaries. It has good, quirky support for functional programming. Mertz looks at how it can be used in a functional paradigm, and points out pitfalls to avoid.

AN ELEMENTARY INTRODUCTION TO THE WOLFRAM LANGUAGE

BY Stephen Wolfram
PUBLISHER Wolfram Media Inc
PRICE £13.69 (free online)
ISBN 978-1944183059
URL **bit.ly/2QU8eZs**

Wolfram is high level, but also suitable for students to get to grips with real-world problems. It pioneered an interactive, functional programming style which lends itself to exploring and solving mathematical problems across many domains.

PROGRAMMING IN HASKELL

BY Graham Hutton
PUBLISHER Cambridge University Press
PRICE £27.22
ISBN 978-1316626221
URL **https://bit.ly/2bI0FpI**

More suited to undergraduate CS than A level, but a great introduction to functional programming in general and Haskell in particular. Hutton explores motivating contexts such as code breaking and the Countdown numbers game.

"HELLO, WORLD!"

Everything you need to know about our computing and digital making magazine for educators

Q WHAT IS HELLO WORLD?

A *Hello World* is a 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 supported by Oracle.

THE MAGAZINE FOR COMPUTING & DIGITAL MAKING EDUCATORS
(Hello World)
Issue 10
Oct 2019
helloworld.cc
WHY TEACH CHILDREN TO CODE?
The role of programming in CS
CYBERSECURITY IN CLASS
Where to get started
PLANNING THE PERFECT OPEN EVENING
Insider's guide inside
SORTING OUT SHAPES
Logo lesson tips
LET IT GROW!
When technology meets the school garden
SEMANTIC WAVES
Ideas for reviewing lesson activities
MATHS & COMPUTER SCIENCE
Fresh ideas and activities for your classroom
PLUS
BRINGING PHYSICAL COMPUTING TO THE CLASSROOM • MAKING NEWS AT SKY • CODE CLUBS FOR TEACHERS
RETRIEVAL PRACTICE & LEARNING • GOING BEYOND THE QUESTION • IMAGINING THROUGH VIRTUAL REALITY

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 100-page magazine is available five times per year.

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 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 or lesson plan? 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