Our Autumn Term edition is once again led by Miles Berry, Professor of Computing Education at the University of Roehampton, with Amy Cartwright, Head of Digital Innovation and Learning at Ibstock Place School adding an article on Teaching Neurodiverg

No images? Click here

SAPIENTIA

From ICT for Education

Everything you need to know about Computing, the curriculum, and the classroom.

Welcome to the autumn term edition of Sapientia, ICT for Education's exclusive newsletter that provides education professionals with thought leadership, an insight into hot topics, and practical guidance on how to implement new technologies and techniques to improve teaching and learning.

This edition features an article by Miles Berry, Professor of Computing Education at the University of Roehampton, that considers the challenges of disadvantaged pupils in life, work and education, and particularly in access to teaching and qualifications in computing.

We also introduce Amy Cartwright, Director of Digital Strategy and Head of Innovation Faculty at Ibstock Place School, who provides information and practical guidance on teaching neurodivergent children to code.

Reviewing the circumstance of disadvantaged children, Berry notes attainment gaps in early years outcomes that widen as pupils get older. In

computing, the average GCSE computer science grade drops from 5.3 in the richest postcodes to 4.2 in the poorest, raising issues about access to further education, apprenticeships and IT professions.

Tackling these types of problems, Berry discusses the options and potential of national policies, as well as opportunities in the classroom and at school level that could improve educational outcomes and prospects for disadvantaged pupils.

Cartwright explains approaches to teaching code to neurodivergent children and argues the case for classes to provide more opportunities for the unique style of learning and the skill sets of neurodivergent children, perhaps by rethinking some of the common pedagogical practices used across school.

Being neurodivergent herself has also allowed Cartwright to look back on her own learning of how to code and bring her experiences into the classroom at Ibstock Place School.

To keep pace with the changes, challenges and opportunities in the primary and secondary education sectors, register for ICT for Education's termly newsletter at https://www.ictforeducation.co.uk/sapientia/ here or email il@ictforeducation.co.uk.

And don't miss our live ICT for Education events, where you can listen to expert speakers, and network with colleagues. Click https://www.ictforeducation.co.uk/ to find out more and register for upcoming ICT for Education events.

Sarah Underwood, Editor - ICT for Education

Tackling disadvantage in computing

By Professor Miles Berry, Professor of Computing Education at the University of Roehampton.

Life, work and education present additional challenges to disadvantaged pupils. There are wide gaps in early years outcomes, performance in key stage 2 SATs and at GCSE between pupils entitled to free school meals and their more affluent peers. These gaps widen as pupils get older. The attainment gap is 20% at age 5, 22% at age 11, and 27% by age 16. It's not surprising that poverty at home affects outcomes at school, but it's worrying that things get worse rather than better over time at school.

In computing, there are big issues around access to teaching and qualifications. In the 20% of most affluent areas, 93% of schools offer GCSE in computer science, but in the 20% least affluent areas, it's just 67%. 16% of pupils in those affluent areas take GCSE computer science, but in the poorest areas it's less than 11%. The average GCSE computer science grade drops from 5.3 in the richest postcodes to 4.2 in the poorest.

The disadvantage gap in provision, uptake and attainment for GCSE computing is particularly worrying as computing is a relatively meritocratic field. Getting on and doing well in the IT industry is much less about accident of birth, what ones parents do or about a private education, but more about an interest in technology, perseverance when things get tricky and a love of problem solving. Yet, if we don't even provide access to the qualifications, or make these accessible and engaging for all kids, then access to apprenticeships or HE, and subsequently to IT professions, will deny these opportunities for social mobility to many.

There are a number of ways to tackle this. Some of these are about national policy, but there's also lots that can be done at school level.

Pupil Premium funding for disadvantaged pupils, and accountability to Ofsted for how this was spent, did help, with the attainment gap narrowing slowly but steadily year on year from 2010 up until the 2019-20 academic year. The good that was done was reversed during lockdowns and home learning in 2020-21, and has been worsening since. Schools receiving Pupil Premium now have a limited catalogue of things on which it can be spent, with most of these being interventions that boost the quality of provision for all. This isn't a bad thing, but it no longer addresses the need to improve progress specifically for the most disadvantaged.

For computing, I'm hopeful that the ongoing curriculum and assessment review might recommend replacing 'narrow, dull and hard' GCSE computer science with a new GCSE covering all of computing, which should make it more practical for schools in less advantaged areas to offer, and would make it appealing to a much broader pupil demographic.

Digital poverty, understood as 'The inability to interact with the online world fully, when where and how an individual needs to', is just one of the challenges facing those living in poverty, but it has a huge impact on any pupil's ability to access curriculum resources outside of school, take charge of their own learning, and engage with programming, digital media and office skills. The previous Labour government provided a highly effective home access scheme, offering computers and internet access to poor families with children in key stages 2 and 3. This was wound up when the coalition government took office, but the new government now has an action plan on digital inclusion, although the first phase of this seems quite modest, given the scale of the problem:

piloting a ‘multi-department device donation scheme to provide re-purposed government laptops to those that need them’ is good, but not enough.

For schools, if pupils studying computing, especially if they’re taking GCSE or A Level computer science, don’t have a computer and internet access, then Pupil Premium can be used to address this. While access to general digital technology isn’t in the Pupil Premium catalogue, the DfE’s rules say that ‘In exceptional circumstances, and where this is necessary to overcome specific barriers to pupil attainment, schools may use this funding on items not included on the list’. Not having access to a computer while studying computing is a pretty specific barrier to attainment.

At the classroom level, we have to be serious about high expectations for all. We can’t take the present attainment gap as a given, lowering our expectations of the grades free school meals pupils will achieve, of how hard they’ll work, or of their ability in our subject. I don’t think knowing predicted grades helps with this. Pupils from poorer homes generally did worse in the SATs than the other kids, and so their predicted GCSE grades are reduced. What worries me is if teachers lower their expectations due to the flight paths they’re given.

Education should broaden pupils’ horizons beyond those of their background. This is especially true for pupils whose home environment reduces access to technology, opportunities to play in and explore the digital world, and understanding of the role of IT and AI in work and society. This means extending the opportunities for playful creativity and digital criticality beyond the requirements of the computing curriculum. It means stretching pupils’ thinking, exposing them to innovative technologies and debating some of the implications of technology in every aspect of life.

As with the gender gap in computing, we can do better with our role models and examples. Big Tech founders and CEOs are the wealthiest of the wealthy, and typically had privileged backgrounds themselves. Many of the great figures from computing history started life with many advantages. One notable exception is Tommy Flowers, born in the East End, son of a bricklayer, and taught as a child to be ‘frugal in everything’. Flowers did an apprenticeship and evening classes before working as an engineer at the post office. He built Colossus, the Bletchley Park computer that cracked the German’s Lorenz cipher; he received little recognition for the work at the time and was left in debt as he’d paid for the valves himself and wasn’t in The Imitation Game. You and your pupils can see a rebuild of his machine at Bletchley’s National Museum of Computing on a school trip. Beyond Flowers and other computing working class heroes, draw on your own former pupils, from backgrounds that your current pupils can immediately relate to, to help lift their aspirations for careers in computing.

I’d like to see all schools offering the GCSE and a far broader demographic taking it than at present. I’d also like to see the achievement gap between the least and most affluent start narrowing again. We need policy to address this, but there’s lots we can do at classroom and school level already.

Book recommendations

Copeland, B. J. (2010). Colossus: The secrets of Bletchley Park’s code-breaking computers. Oxford University Press. A great collection of chapters about this ground-breaking code cracking computer, including a brief history of cryptography, Tommy Flowers own account of building Colossus and Tony Sale’s story about rebuilding it.

Hodges, A. (2014). Alan Turing: The Enigma: The Book That Inspired the Film” The Imitation Game”. Princeton University Press. Readable biography of

one of the heroes of computer science as a discipline, responsible for the foundational notion of the Turing Machine, the still relevant AI 'Turing Test' and so much of the Bletchley Park work in World War II.

Register to meet and hear Miles speak at the ICTfE Solihull on November 7th at Cranmore Park Plus a further opportunity at the ICTfE Newcastle Conference on December 5th at St james' Park

Professor Miles Berry

Miles Berry is Professor of Computing Education at the University of Roehampton. Before joining Roehampton, he spent 18 years in schools, including a period as a head teacher. He has contributed to a wide range of computing projects, including the computing programmes of study in the National Curriculum, Barefoot Computing and Switched On Computing. He serves on the boards of Computing At School, the BCS Academy of Computing, and the National Centre for Computing Education, and is a regular keynote speaker and international consultant on curriculum and professional development. He is @mberry on Twitter and find out more on milesberry.net

Hear Miles speak at ICT for Education Conferences at Solihull (November 7th) and Newcastle (December 5th).

Teaching Neurodivergent Children to Code

By Amy Cartwright, Director of Digital Strategy, Head of Innovation Faculty, Ibstock Place School

The current zeitgeist in Computing education is a complex one and I have found myself spending a lot of time working on issues like improving gender balance and policing AI, all the while overlooking

the core principle of my job, teaching children. So, for a while now, I have been attempting to build a stronger foundation by focusing on classroom practice and computing pedagogy.

As part of this refocus, I have been thinking a lot about the neurodivergent children in my classroom, particularly teaching them how to code. I am neurodivergent myself. I have an ASD level 1 diagnosis, which would have been called Aspergers if I was diagnosed earlier in life. It was a late diagnosis and it has allowed me to look back on my own experiences of learning how to code with a little more empathy and understanding.

The media likes to equate neurodivergence with computer science (anyone seen Mr Robot?) and it's easy to see why. Programmers use logical thinking and pattern recognition, they aim for a flow state to enhance their hyperfocus. They need to be good problem solvers and the machine they need to understand is entirely literal.

So, do we think that neurodivergent children are drawn to the subject because of these commonalities? Perhaps. Do these commonalities ensure success? No.

The reality of a neurodivergent brain is best expressed as a spiky profile that can help us to understand the different ways that neurodiversity manifests in each individual. As teachers, we already know that not every child with a neurodivergent brain is a going to be a naturally gifted programmer and sometimes we even see some of these traits working against children in our classrooms. We see autistic children struggling with the inflexibility of a classroom, we see children with ADHD unable to access a flow state and we see children with dyslexia finding programming syntax challenging.

The simple fact is, learning how to program in a classroom setting can be particularly challenging for a neurodivergent child.

When I was learning to code, I was set a task by my teacher that was to replicate a famous painting. I chose Kandinsky's 'Circles in a Circle' and produced a pretty accurate replica. I was working in a program called Open Frameworks that uses a co-ordinate system for placing shapes on the screen, similar to turtle. My program was massive, endless lines of sequenced shapes with no conditions, loops or subprograms in sight.

At the time of this task being set, we had surpassed the programming requirements for GCSE so I was a competent programmer. This piece of work did not demonstrate that skill at all. What happened? Literal thinking. Most of my peers went away and chose an artwork by Mondrian or Warhol or something else that was heavily pattern based with some repetition. They understood that the task had an objective beyond the replica. They understood that the point of this would be to demonstrate and improve their programming skills. I did not. Instead, I spent hours and hours tweaking the co-ordinates for each shape aiming for the perfect replica.

This wasn't a failing on the teacher's part or on mine. It's an example of two people communicating in two different ways. As teachers we have to learn how to communicate so that all of our pupils understand. I find that it helps if I am transparent about why we are doing tasks.

So, when asking a pupil to: *"Produce a replica of a piece of modern Art"*

I might add something like this: *"This is a fun way to practice some of the more complex structures we have been working on in class. When selecting your artwork, make sure to consider how you might make best use of the programming skills*

you have learnt to date. Your code should include loops and subprograms"

The task is still quite an open one. By providing these additional instructions, I haven't made the task easier or less creative. All my pupils value this level of transparency and, importantly, there is less chance of a misunderstanding from some of the more literal thinkers in the room.

Another of the significant difficulties I experience is with executive functioning, specifically task switching. I found this difficult in school, needing to move from lesson to lesson, usually only just getting into a task before needing to pack up and move on. Now, as an adult, when I am working on a project, I will make more progress when I have a large chunk of time and the opportunity to become engrossed. I often think of my brain like a microcontroller. Switching between different activities quickly is just not possible. I require a hard reset each time and new code to be uploaded.

You can see the challenge in a classroom setting here. If we want to limit task switching and provide opportunities for flow, it goes against the preferred teaching method of many; small quick tasks focused on developing specific skills that are punctuated with teacher exposition and questioning. There are likely teachers who would argue that the flow state is not the desired outcome in a classroom but for some neurodivergent children, not being able to access this state might be a hinderance to their progress.

One useful strategy I have found is flow lessons. These are essentially flipped learning with a twist. I schedule a few of these sessions in per half term. The pupils arrive for this lesson having been fully prepared the lesson before. They will either be working on a series of smaller related tasks or one long form task that will take a whole lesson. I allow pupils some autonomy over how they work in

these sessions, and I respect their choices. Some will choose to work independently in a quiet corner of the classroom with limited interaction with me whereas some will choose to check in frequently.

These flow lessons only really work if the task that is set is achievable within one lesson. We all know the frustration of being pulled out of a flow state when you haven't completed something. It's also important to ensure that tasks are still appropriately scaffolded to avoid overwhelm, particularly when teaching children who have not been coding for long.

It's not only the classroom that is not optimised for flow. I teach the OCR A Level specification and have noticed that some pupils have difficulty with the development section of the NEA. It requires the pupil to document the code as they progress, adding screenshots and commentary into a written report. For someone that relies heavily on a flow state, this can be a really challenging task.

One solution that has worked for some is to use the 'Win + PrtScn' shortcut as they are coding. This allows them to take frequent screenshots that will be saved in chronological order without needing to leave the IDE. This allows them to separate out the tasks of coding and documenting as they will be able to add these screenshots to their report at the end of their coding session along with annotation and commentary. Of course, this does require the pupil to be well organised, which can be a difficulty for many, but I do find that reducing the need for task switching allows the pupil to progress with less frustration.

This article is far too brief to consider all the unique perspectives of a neurodivergent child and all the challenges presented when learning how to code. Instead, I hope it has prompted some further thought into how we might adapt our teaching when working with neurodivergent children. There are certainly a large number of highly successful

neurodivergent people working in Computing, but I don't think that we can call that a success story. Classrooms need to be catered more towards providing opportunity for their unique style of learning and skill sets. This will require rethinking some of the common pedagogical practices that you see in other classrooms around your school.

Book Recommendations

1. The Autistic Brain – Temple Grandin – If you value science backed reasoning and would like to know more about both the Autistic experience and what we know about the Autistic brain on a biological level, you will find this book interesting.

2. Camouflage – Dr Sarah Bargiela – This will help you to better understand how autism presents in girls. This has been so misunderstood for so many years that I recommend all teachers read this. It's a delightful graphic novel that can be read in one sitting. It will certainly stay with you for much longer!

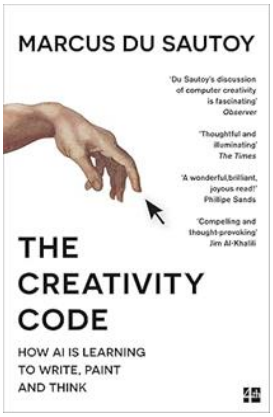
Amy Cartwright

Amy Cartwright is a creative technologist, dancer, artist and teacher. Following the completion of her MA in Computational Art in 2017, Amy has worked as a freelance dance artist and continued her research into robotics and dance. Alongside this, for the past three years, she has worked at Ibstock Place School under two titles: Head of Computing and Head of Digital Innovation and Learning. She has revitalised the Key Stage 3 curriculum and introduced a GCSE in Computer Science. She has also been at the forefront of the schools move into digital learning, working on digital pedagogy across faculties and advising on updates to the digital infrastructure.

Further reading

Miles Berry, Professor of Computing Education at the University of Roehampton recommends

Groß, B et al (2018) Generative Design: Visualize, Program, and Create with JavaScript in p5.js, Princeton Architectural Press



p5.js is an online, interactive coding environment for producing gorgeous digital art. It captures Resnick's vision of low floors, wide walls and high ceilings, and is a great way to connect coding with more engaging, creative work.

du Sautoy, M (2019) The Creativity Code: How AI is learning to write, paint and think, Fourth Estate.

Oxford mathematician Marcus du Sautoy discusses how AI can do what we might think of as creative work, including in his own field. This predates the rapid advances we've seen in generative AI, but the accessible approach means this is still a top recommendation.

Learn, Share, Enjoy

ICT for Education can help you get the most out of technology in your classroom and in your school. Our free to attend conferences and seminars provide relevant, innovative, informative content delivered by experienced, knowledgeable, respected speakers able to relate to and understand the challenges faced by those responsible for giving learners the best opportunities in life.

Attend a Conference

Salford 9th October

Solihull 7th November

Newcastle 5th December

SAPIENTIA
From ICT for Education

Forward

ICT for Education - 2025

https://www.ictforeducation.co.uk/

You are receiving this email as you are on our database and may also have attended one of our ICTfE Conferences or Seminars.

Preferences | Unsubscribe