Section 1:
Computing in context

Programming and mathematics: insights from research in England

Dame Celia Hoyles (University College London, UK)

Hoyles, C. (2021). Programming and mathematics: insights from research in England. In Understanding computing education (Vol 1). Proceedings of the Raspberry Pi Foundation Research Seminar series.

Available at: rpf.io/seminar-proceedings-2020

Raspberry Pi

Section 1: Computing in context

Programming and mathematics: insights from research in England

Dame Celia Hoyles (University College London, UK)

I start this short piece by providing a glimpse of the way computing was introduced in schools in England, culminating in 2014 with the introduction of a new statutory primary national computing curriculum for students aged 6 to 16 years. One key landmark on the way was the influential report from the Royal Society, *Shut Down or Restart* (The Royal Society, 2012). Then came the second Royal Society report, *After the Reboot: Computing Education in Schools* (The Royal Society, 2017), which recognised the importance of the teacher for the success of the curriculum initiative and led to the setting up of the National Centre for Computing Education, NCCE, specifically to support the teaching of computing.[1]

The computing curriculum included, as a key aspect, that students should design, build, and debug programs. My main personal concern was how programming, as well as being part of computing, might also fit with the rest of the curriculum, with particular reference, given my background, for mathematics. Could this curriculum innovation of computing be harnessed for the benefit of all subjects?

To address this question, I want to provide an outline of the history of programming and mathematics, which for me had its roots in innovations from MIT in the 1980s, and the vision of Seymour Papert for the development of Logo, as a language for learning. This is when I became personally convinced of the potential for programming in mathematics teaching and learning; as a teacher I experienced the 'buzz' of a classroom where the learners were actively engaged in exploring and discussing mathematical ideas through programming them. At the same time, Papert proposed his theory of constructionism, that proposed that learning tended to be effective when making an artefact that was personally or socially meaningful, could be shared with others, reflected upon, and debugged (see for example (Kafai & Resnick, 1996)).

In this early work, the notions of powerful ideas and design were stressed; that is a *well-designed* constructionist activity should have *personal* meaning and *emotional* connection with learners and empower them in some way, put simply so they could do something that before they were unable to do. Such ideas had deep resonance for me as so much of mathematics is not experienced as personally meaningful, just seen as simply a 'dance of symbols' without underlying structure and linked to little emotion except anxiety, feeling stupid and 'not getting it'.

During this time, a group of us set up the LogoMathematics group which met regularly with participation from across the world, leading directly and indirectly to publications over the subsequent 50 or more years (see for example, (Papert, 1972; Hoyles & Noss, 1992; Noss & Hoyles, 1996; Monaghan, Trouche, & Borwein, 2016)).

After setting the scene I will describe the ScratchMaths (SM) project, (now called *UCL ScratchMaths*), the latest research project

[1] http://teachcomputing.org

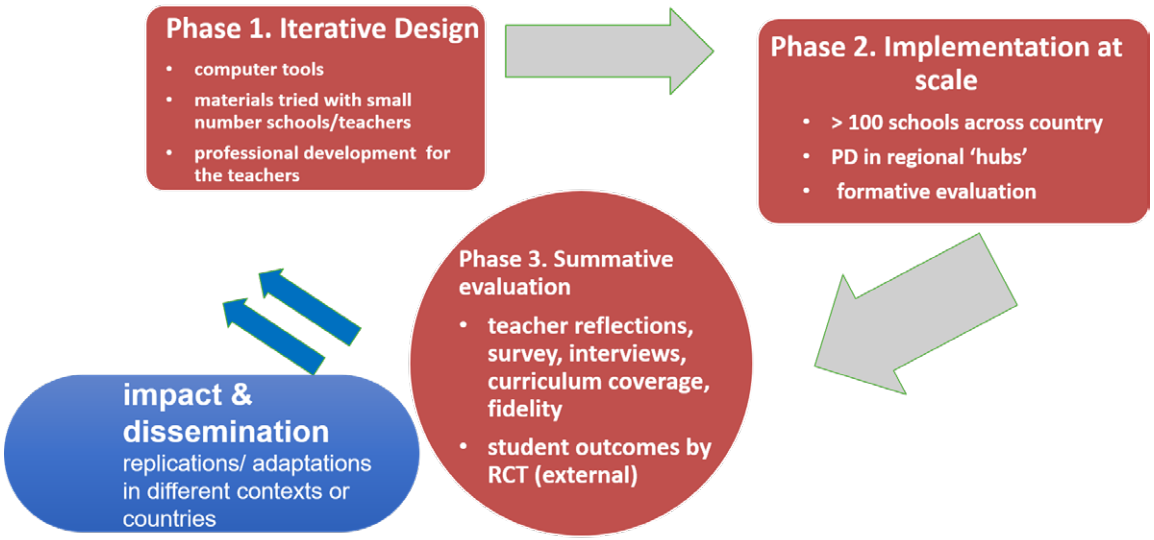


Figure 1. Phases of the ScratchMaths research

in which I have worked that looked at the programming/mathematics interface. The UCL ScratchMaths designed and implemented a longitudinal two-year intervention at the intersection of mathematics and computing, targeted for 8- to 11-year-old students in English schools and involving programming in Scratch (Noss, Hoyles, et al., 2020).

The phases of the ScratchMaths research are shown in Figure 1. Much of the effort of the ScratchMaths team took place in the first phase, the Iterative Design Phase, where the we worked with a small group of schools to iteratively design computer tools and student/teacher materials and pilot them in the schools, along with a programme of professional development for the teachers.

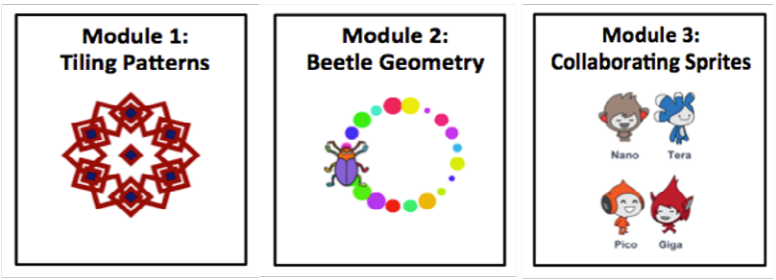
Our team was interdisciplinary[2] with expertise in mathematics education, computing, and design, and we worked closely with teachers to iteratively develop our original designs. We aimed to foster *mathematical thinking*. *This can be defined as an awareness and appreciation of mathematical structure, the articulation of coherent explanations for outcomes and the reasoning behind them, and being comfortable and fluent with the formal expression of relationships*.

To pursue this aim, we developed student and teacher curriculum support materials organised into six modules, three to be taught per year, involving about 20 hours teaching. The modules can be considered as *microworlds*, designed to provoke engagement with key ideas in mathematics and in computing. (For background on microworld development, see (Hoyles, 1993), and the recent contributions of Chronis Kynigos to the Mathematics Knowledge Network lecture series.[3])

[2] Professor Dame Celia Hoyles (Mathematics) and Professor Richard Noss (Mathematics) - UCL Knowledge Lab, Professor Ivan Kalas, (Computing) - Comenius University, Bratislava, Slovakia Dr Laura Benton (Computing) and Piers Saunders (Mathematics) - UCL Knowledge Lab, Prof Dave Pratt (Mathematics) - UCL Institute of Education

[3] http://mkn-rcm.ca/online-seminar-series-on-programming-in-mathematics-education

Year 5 (9-10 yrs) – Computing focs (20+ hours)



Year 6 (10-11 yrs) – Mathematics focs (20+ hours)

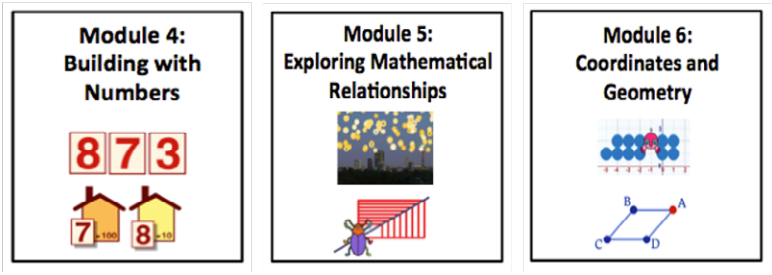


Figure 2. Overview of UCL ScratchMaths

Figure 2 shows a summary of the six UCL ScratchMaths microworlds produced.

All the materials are freely available, now updated to Scratch 3.0, through the UCL website.[4]

The ScratchMaths team took as the following components of computational thinking[5] derived from a large number of rather similar definitions and resources available at that time. (For background, see (Benton et al., 2017)) :

- *Abstraction*: seeing a problem and its solution at many levels of detail
- *Algorithms*: thinking about tasks as a series of logical steps
- *Decomposition*: understanding that solving a large problem can involve breaking it down into smaller problems
- *Pattern recognition*: appreciating that a new problem is likely to be related to other problems already solved
- *Generalisation*: realising that a solution to a problem can be made in ways that can solve a range of related problems

In Phase 1, it also became clear that we needed an explicit pedagogic framework for ScratchMaths to facilitate our future work, and we devised one with the teachers in the four design schools referred to as the "five Es", with each E derived from a wealth of prior research in mathematics education about effective pedagogy:

[4] http://www.ucl.ac.uk/scratchmaths

[5] For an up-to-date- summary of definitions and research on Computational Thinking, see Paul Dryvers lecture as par of the Mathematics Knowledge Network lecture series, available at http://mkn-rcm.ca/online-seminar-series-on-programming-in-mathematics-education/.

Explore: investigate, try things out yourself, debug in reaction to feedback.
Envisage: have a goal in mind, predict the outcome of the program before trying on the computer.
Explain: explain what you have done, articulate reasons behind your approach to yourself and to others.
Exchange: collaborate and share, try to see a problem from another's perspective as well as defend your own approach and compare with others.
bridgE: make explicit links to the mathematics curriculum.

In relation to the last E, *bridgE*, we had learned from our earlier research in programming and mathematics that often we had assumed connections between these two fields would be made, only to find that this was not the case. All was too implicit and assumed. The five Es framed the professional development programme we devised (two days per year), a programme that was a critical part of the SM intervention and the planned classroom implementation.

Alongside the design phase the SM team planned for *Phase 2*. *Implementation at scale*. We signed up to the project over 100 schools across England grouped around seven regional hubs that would form the focus for professional support and formative evaluation. At this point, an external independent evaluator was appointed by the funders who undertook to assess the project in terms of its effect on scores of computational thinking and of mathematics. The results of the ScratchMaths intervention are reported in full in the evaluation report.[6] We note that ScratchMaths had a positive and significant impact on student computational thinking (CT), as reported by the evaluator using a randomised control trial methodology with 111 schools across England and measured by a test of computational thinking designed and administered by them at the end of the first year of the intervention. We also note the important results that this positive effect was particularly evident among educationally disadvantaged students. There was no evidence of any interaction between the impact of SM on CT test scores and gender: thus girls and boys appeared to engage with SM to a similar extent, an outcome that is particularly important in view of the finding persistent in the literature that girls tend not to be so engaged in computing as boys.

However, there was no impact of SM on mathematics attainment as measured by the evaluators on the basis of the student results in the statutory national mathematics test, Key Stage 2 test taken by all 11-year-old students in England. As a way to seek to explain these findings, I called on the notion of *fidelity of implementation* (see (O'Donnell, 2008) for a review of *Defining, Conceptualizing, and Measuring Fidelity of Implementation*), and how in our study fidelity appeared to have been negatively influenced by the high-stakes testing in mathematics in England leaving little room for innovation in classrooms for 11-year-olds. These tests involved a formal paper and pencil mathematics test and are used to rank schools and teachers so much time is spent reviewing and revising, so teachers found little resource to devote to ScratchMaths.

Finally, I want to dwell a little on the final and still ongoing phase of the ScratchMaths project, concerning how it has been disseminated and its impact on other projects in England and internationally. The materials have been used in a great many countries across Europe, and beyond. I note in particular that the ScratchMaths project has been followed up in New South Wales in Australia (see Holmes, Prieto-Rodriguez, et al.,

[6] https://educationendowmentfoundation.org.uk/projects-and-evaluation/projects/scratch-maths/.

2018) where it continues to be widely adopted. Also of note is the nationwide project that took place in Spain (INTEF, 2020), part of which concerned a replication of the ScratchMaths project along with assessing its impact. I translated one finding from this report that was of particular relevance: namely that *"the results show that it is possible to include programming activities in 5th grade in the area of mathematics, so that students not only learn to program and engage in computational thinking, but also improve the development of their mathematical competence greater than their colleagues who have worked in this same area using other types of activities and resources not related to programming."*

I would like to end by reflecting on how the ScratchMaths research might be improved or updated in future work, not least as teachers are becoming more confident and competent in their understanding of computational concepts, in teaching them and in using them to explore mathematical ideas through programming. So I present some personal thoughts about our project and its limitations with the wisdom of hindsight, and pose some research challenges that might be interesting for others in the community to address. For example, the need to:

1. Develop more *nuanced, rigorous, and targeted assessment instruments of student and teacher content knowledge in mathematics, mathematical thinking, and in computing* to be administered as post-tests following engagement in each of the ScratchMaths microworlds and as delayed post-tests several months later, rather than use the standard national tests as adopted in the evaluation of UCL ScratchMaths.
2. Research in more detail the actual practices in classrooms to include documentation of teacher and student interactions and output, in order to provide detail of classroom implementation and how far the pedagogic framework was enacted. In particular, such research might provide some explanation of the outcomes reported for UCLScratchMaths in relation to socially disadvantaged students and girls as mentioned above, taking as a starting point the idea of fidelity while recognising the 'chaotic' nature of real classrooms, teacher practices, and policy demands.
3. Develop a more detailed description of the *nature and content of the professional development* that is undoubtedly needed prior to successful implementation of the ScratchMaths intervention.

At the time UCL ScratchMaths was conceived and operationalised, computing was very new in England. Teaching and learning has been transformed in the intervening years, not least as much education has moved online as a result of the coronavirus pandemic, and the magnificent efforts of the NCCE to support the teaching of computing across the country. Teachers and students have undoubtedly become more fluent in working online in general and in programming in particular. One might expect that the integrity of the SM materials would remain constant, given the principles of design on which they were based while its implementation would be less challenging in these changed circumstances. But this is a matter for further research.

References

Benton, L. Hoyles, C., Kalas, I & Noss, R. (2017). Bridging primary programming and mathematics: preliminary findings of design research in England. In *Digital Experiences in Mathematics Education*, pp 1- 24

Holmes, K. Prieto-Rodriguez, E., Hickmott, D. & Berger, N. (2018). Using coding to teach mathematics: Results of a pilot project. *Proceedings of the 5th International STEM in Education Conference*. Brisbane.

Hoyles, C. (1993). Microworlds/Schoolworlds: The transformation of an innovation. In Keitel, C., Ruthven, K. (eds) *Learning from Computers: Mathematics Education and Technology*. NATO ASI, Series F: Computer and Systems Sciences, 121, 1-17.

Hoyles C. and Noss, R. (1992). (eds) *Learning Mathematics and Logo*. Cambridge MA: MIT Press.

Kafai, Y., & Resnick, M. (1996). Constructionism in Practice: Designing. Thinking, and Learning in a Digital World. New York: Taylor & Francis Group.

INTEF (2020). La Escuela de Pensamiento Computacional (School for Computational Thinking (The School for Computational Thinking). Instituto Nacional de Tecnologías Educativas y De Formación Del Profesorado (National Institute of Educational Technologies and Teacher Training) Final Report . Available at: https://intef.es/tecnologia-educativa/pensamiento-computacional/

Monaghan, J., Trouche, L., Borwein, J. M. (2016). Tools and Mathematics. Springer

Noss, R. and Hoyles, C. (1996) *Windows on Mathematical Meanings: Learning Cultures and Computers*. Dordrecht: Kluwer Academic Publishers.

Noss, R., Hoyles, C., Saunders, P., Clark-Wilson, A., Benton, L., and Kalas, I., (2020). Making constructionism work at scale: the story of ScratchMaths: In Holbert, N., Berland, M., Kafai, Y. *Designing Constructionist Futures: The Art, Theory, and Practice of Learning Designs*, Cambridge, MA: MIT Press.

O-Donnell, C. (2008). Defining, conceptualizing, and measuring fidelity of implementation and its relationship to outcomes in K–12 curriculum intervention research. *Review of Educational Research*, 78, 1, 33-84

Papert, S. (1972). Teaching children to be mathematicians vs. teaching about mathematics. *International Journal of Mathematics Education in Science and Technology*, 3, 249-262.

The Royal Society (2012). *Shut down or restart? The way forward for computing in UK schools*. Available at: https://royalsociety.org/topics-policy/projects/computing-in-schools/report/

The Royal Society (2017). *After the reboot: computing education in UK schools*. Available at: https://royalsociety.org/~/media/policy/projects/computing-education/computing-education-report.pdf

Raspberry Pi