Hoyles, C., Gadanidis, G., Kwon, O.N., Modeste, S., & Prieto-Rodriguez, E. (2023). Mathematics Education Incorporating Coding: Practical Challenges and Opportunities. In Online Proceedings of the *Coding, Computational Modelling, and Equity in Mathematics Education Symposium*, St. Catharines (Canada), April 2023.

Mathematics Education Incorporating Coding: Practical Challenges and Opportunities

Celia Hoyles[1], George Gadanidis[2], Oh Nam Kwon[3], Simon Modeste[4], & Elena Prieto-Rodriguez[5]
[1]UCL Knowledge Lab, University College London, London, U.K.;
[2]Western University, Ontario, Canada, ggadanid@uwo.ca;
[3]Department of Mathematics Education, Seoul National University, Korea;
[4]Institut Montpelliérain Alexander Grothendieck, University of Montellier & CNRS, France, simon.modeste@umontpellier.fr;
[5]School of Education, University of Newcastle, Australia, elena.prieto@newcastle.edu.au;

The Panel

The Panel was introduced by Dr Ana Isabel Sacristán Rock, Centre for Research and Advanced Studies (Cinvestav-IPN), Mexico.

This panel aimed to complement the research-focused panel held on the previous day, which focused on the question: *How do computational thinking and mathematical thinking interact in terms of knowledge, ways of thinking, and competencies?*

The following abstract provides an overview of the session along with a structure for the illustrative texts and videos presented by panelists, which are summarized at the end.

Abstract

Coding or programming is ubiquitous across the world. But what coding means, how it is learned and developed and how it is exploited as a tool to explore topics in different subject areas varies enormously across jurisdictions: for example, coding can be introduced and developed as part of a school computing curriculum, as part of the school mathematics curriculum or more informally within out-of-school clubs. These different structural organizations inevitably serve to define what happens in practice in schools, shapes how students develop coding skills and learn key coding concepts, informs how teaching might be enhanced through coding, and ultimately how coding could be exploited outside of computing as a tool to think with and explore mathematics.

Each panelist set out to touch on these issues with a particular focus on the interactions between mathematics and coding *in practice* in their country. The discussion aimed to tease out the challenges, risks and opportunities of integrating coding into mathematics classrooms while addressing questions such as: Why should coding be incorporated in mathematics classrooms? What is the specificity of coding in mathematics and the links between coding and mathematics? What can coding bring to mathematics in terms of new content or new ways to support mathematics teaching and improve learning? Which mathematical topics are most aligned to the incorporation of coding and why? What are the links between coding and algorithmics, applied mathematics and reasoning?

The panel comprised a chair and four invited panelists from four different countries with very different curricular structures.

Chair: Prof Dame Celia Hoyles, United Kingdom

Panelists:

- Dr Oh Nam Kwon, Korea
- Dr Simon Modeste, France
- Dr Elena Prieto, Australia
- Dr George Gadanidis, Canada

Reactor: Prof Richard Noss, UCL Knowledge Lab, University College London, London, U.K.

The duration of the panel was 1.5 hours, following an introduction by Dr Ana Isabel Sacristán Rock. It comprised of:

- Celia Hoyles' introduction to the topic, the panel, and the panellists (5-10 min).
- Each panelist then briefly described their country's positioning of coding or programming in the school curriculum and any links with mathematics, illustrating their perspective on classroom implementation in their country with short videos, photographs or with student work (10 minutes each).
- The Chair and the panelists were then invited to provide short reactions to the presentations 5 mins each.
- Richard Noss then made his reaction to the panel session.
- Finally, questions were taken from the audience face-to-face or online.

Below is a brief summary of each of the presentations.

Mathematics Education Incorporating Coding: Practical Challenges and Opportunities in Korea.

Oh Nam Kwon, Seoul National University, Korea

I explore the integration of coding into Korea's mathematics education, focusing on the AI Mathematics curriculum introduced in high school in 2020. This initiative, developed due to increasing societal demand for AI knowledge, emphasizes the relationship between AI and mathematics, data representation, categorization, prediction, and optimization, incorporating key mathematical concepts like vectors and matrices. Furthermore, I investigate into how coding, specifically block and text coding, is woven into this curriculum, aligning with Korea's national education strategy. This is exemplified through the classroom practices of Mr. Oh, a high school teacher who uses digital tools like Askmath AI Lab and Algeomath. Mr. Oh's experience highlights the importance of integrating mathematical concepts with AI applications, enhancing students' interest in both areas and preparing them for future technological challenges and STEM careers. The integration of AI and coding into mathematics education not only addresses practical challenges but also opens up significant opportunities for enriching students' learning experiences, blending traditional mathematics with modern technological applications. The innovative approach of integrating AI and coding into mathematics education in Korea, particularly through hands-on examples like those in Mr. Oh's classroom, represents a forward-thinking shift in educational practices. This paradigm shift not only makes mathematics more engaging and relevant for students but also bridges the gap between theoretical knowledge and real-world applications. This integration thus serves as a model for other educational systems worldwide, illustrating the benefits of adapting traditional curricula to include emerging technological trends.

An Example From an 'Ordinary' Situation in Grade 9 in France

Simon Modeste, University of Montpellier, France. *simon.modeste@umontpellier.fr*

The aim of this presentation was to present an example from France of coding in mathematics in middle school (in grade 9), illustrated with a video.

General Curricular Context

To situate the context, Table 1 presents a synthesis of the current curricular situation in France, concerning mathematics and computer science. In primary school, computer science has been introduced as a subject since 2015 (note that in France, before grade 6, there is one teacher

for all the subjects). In middle school, computer science has been introduced inside two subjects: *technology*, and *mathematics*, the latter in a specific theme called "algorithmics and programming", and the programming language is Scratch. In high school, computer science is an independent subject with dedicated teachers, but notably, despite this, mathematics has kept the specific theme "algorithmics and programming" with Python as the programming language.

Table 1. Synthesis of the presence of "coding" in French curricula, in mathematics and computer science.

Grades	**School level**	**Organization/division of School subjects**		**Language**
1-5	Primary school	Mathematics / Computer Science (one teacher for all subjects)		Scratch Jr Robots
6-9	Middle school	Computer Science (distributed into)	Mathematics ("algorithmics and programming")	Scratch
			Technology	Scratch / Robots
10-12	High school	Computer Science ("major")		Python & others
		Mathematics (including "Algorithmics and programming")		Python

Although France has an ambitious curriculum, there are important differences between the prescribed, implemented, and attained curricula.

The implementing of the curricula depends on:

- the reality of the material and temporal contexts of teaching,
- the variety of the experiences of teachers with coding, and the weak in-service teacher education developed for computer science,
- The balance and links that are needed to make between managing balance and links between Computer Science and Mathematics learning contents.

In practice, "attained curricula", we can observe.

- a heterogeneity of the teachers' practices (including not teaching coding at all),
- difficulties to achieve the learning objectives of this curriculum (at the end of primary school, the end of middle school, and the end of high school),
- an important heterogeneity of experiences and knowledge in coding for students at the end of the (mandatory) scholarship.

For the following and the example presented, we will focus on mathematics in middle school. In the mathematics curriculum for grades 7 to 9 (called "cycle 4"), the 5th theme (among 5) concerns "algorithmics and programming." Table 2 reproduces the content of the theme, where elements are highlighted that serve to indicate the 'thrust' of the curriculum.

Table 2. The part of mathematics curriculum of cycle 4 (grades 7-9) concerning "Algorithmics and Programming."

Theme E - Algorithmics and Programming
In Cycle 4, students are introduced to programming by developing a few simple programs in a project-based approach, without aiming for expert and exhaustive knowledge of a particular

language or software. By creating a program, they develop programming methods, revisit the notions of variables and functions in a different form, and practice reasoning.

Examples of possible activities: games in a maze, Pong, Battleship, Nim, Tic Tac Toe, Exquisite Corpse.

End of cycle expectations

- Write, develop and execute a simple program.

Write, develop and execute a program
Knowledge - notions of algorithm and program. - notion of computer variable. - triggering of an action by an event. - sequences of instructions, loops, conditional instructions. Associated competences - write, develop (test, correct) and execute a program in response to a given problem

Example: An "Ordinary" Classroom of Mathematics in Grade 9 (15 years old)

Simon then introduced the example studied for this presentation. The mathematics teacher, Damien, is an ex-engineer with a background in programming and computer science. He places a specific focus on coding, "independently of mathematics contents" and based on video-games design. One constraint is that his teaching must take place in a computer room (figure 1), where there are not enough computers for his 9th grade class, and it is only possible to teach a half-group. So, Damian separates the class into two groups: one group works autonomously on mathematics exercises in the center of the classroom, while the other working in pairs explore pre-filled Scratch project files, containing instructions as to what they must do along with some help (figure 2).

Figure 1. Configuration of the classroom, with computers around the room and tables in the center.

The sequence is organized into 6 activities (planned to take over 3 or 4 one-hour-lessons and homework), with the goal of making students design a videogame as a final project:

1. Moving a car using coordinates
2. Moving the car with the arrows keys
3. Make a "realistic" U-turn
4. Manage random events
5. Manage obstacles with an associated point/scoring system
6. Make the background move

The video presented and the lesson observed concerned activity 2: making a car move using the keyboard. This activity has 3 steps. The first is to make the car move with the keyboard, which as seen in figure 3 (using an infinite loop, as expected from the teacher's instructions). Then, the students are asked to control the speed of the car and make it fixed at 200 pixels per second. And finally, students have to make the car accelerate to 400 pixels per second, while the "d" key is pressed.

Figure 2. Example of a Scratch project, with pre-filled programs to be completed, and a tutorial programmed by the teacher with the Scratch cat presenting instructions and giving explanations and help.

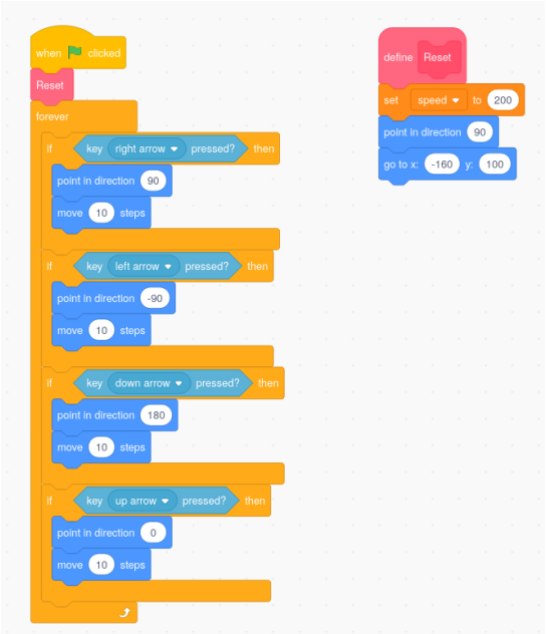

Figure 3. Example of program making the car move in the four directions using the keyboard. The students are expected to make the car move with a speed set at 200 pixels/second, and then it double to 400, while the "d" key is pressed.

The edited video was presented to show two girls working on the task and interacting with the teacher. In the video, the girls had produced a solution as in figure 3 but struggled to set the speed at 200 pixels per second. Seeing that many students had similar difficulties, the teacher gave a collective presentation at the board, where he explained the principle of building a "frame

by frame" move of the car ("like in the movies") where the speed can be controlled. Notably, mathematical contents 'appeared' at this moment, because proportionality is involved through the property "speed = distance / time" and calculating distance should be completed after each step of 1/10 seconds. When the two girls come back to work, they tried to simply copy the formula given by the teacher into their program without trying to understand it. They copied "add 1/10 to speed" (speed was a variable) instead of "wait 1/10 seconds". They then observed that it did not produce what they expected: they ran the program and observed that the car did not move quicker, although the displayed value of the speed variable increased continually. After many trials and discussions with the teacher, they were able to debug their program. They finally managed to have a correct instruction for moving at a given speed and placed it in the right place in their program. Similarly, after trial, errors, and interventions of the teacher, they also solved the last task, namely doubling the speed.

Conclusion: Some Observations and Topics for Discussion, From This Video

About syntax and semantics:

- We see the important role, in coding, of the retroactions of the software in fostering an experimental approach.
- But we see also the risk of purely syntactic work from the students (kind of "push-button" effect, copying solutions…), so we ask ourselves "what did they really understand"?

About the didactical contract:

- It seems that it was not clear for these students what was expected? Was it a (formal) code, or a solution to a problem? And how is the solution validated? By who?
- We can notice the important role of the teacher, and his interactions with the students. Even if sometimes, the teacher took charge, the corrections of the program were left to the students.

Finally, about mathematics and coding:

- We presented here an interesting example of mathematics that appeared *in* and *for* coding, in particular with the speed management, which is different from the classical view of coding *for* mathematics.
- But we can observe the difficulty for students to leave the computer and its short action-feedback loop, to go back to paper-and-pencil work while coding. We see that this is difficult for the two students observed. However, it seems necessary to encourage them to do this, and to think about what they need to solve mathematically regarding their coding problem.

Practice-Focused Discussion Panel on Mathematics and Computing: A View from Australia.

Dr Elena Prieto, School of Education, University of Newcastle, Australia
elena.prieto@newcastle.edu.au

In September 2015, the national Australian Curriculum was officially endorsed. This curriculum includes the new Digital Technologies (DT) subject, within the Technologies learning area, which is focused on the teaching of *"computational thinking and information systems to define, design and implement digital solutions."* (ACARA, 2018b). The document stated that this subject was to be mandatory for all Australian students from Kindergarten to Year 8 and available as an elective for Year 9 and 10 students. However, the landscape of Australian

schooling is decentralized and complex, and even though the Australian curriculum places computational thinking within the Digital Technologies curriculum, in New South Wales, its most populous state, it appears within the Science and Technology syllabus (NSW Education Standards and Assessment Authority, 2017).

Interestingly, neither of these syllabi incorporate computational thinking within mathematics, even though computational thinking is closely tied to mathematical thinking (Selby & Woollard, 2013). Additionally, very few teachers, particularly in the primary school setting, have had any formal schooling on computational thinking or coding and there were concerns that they possessed pedagogies to teach them authentically.

Presumably to facilitate incorporating the skills and content, the NSW Education Standards Authority (NESA) prepared the *Coding and computational thinking across the curriculum guide* for teachers, which aimed to develop algorithmic and computational thinking skills to better enable students and teachers to reach a coding goal. The guide highlighted the areas where computational thinking can be applied within the existing NSW K–8 syllabuses and contained activities and links to resources organized by stages of learning and learning areas. This guide has, however, been discontinued and can no longer be found in the NESA website as the new priorities have come in focusing on Aboriginal and Torres Strait Islander histories and cultures, Asia and Australia's engagement with Asia, and Sustainability.

In this climate, my work has focused on the integration of coding and computational thinking into the mathematics curriculum. In particular, over the past few years I have been working with Celia Hoyles and Richard Noss on the implementation in Australia of ScratchMaths. ScratchMaths is a two-year computing and mathematics-based curriculum for Key Stage 2 in the UK (equivalent to Years 4 and 5 in Australia).

For the Australian implementation, we conducted professional development with teachers for roughly 8 weeks, commencing with a 2-day professional development workshop and ending with a final showcase where teachers shared their experiences and samples of students' work. We also offered support during the interim classroom implementation period. The aim of the project was to explore participant teachers' perceptions of their ability to facilitate students' learning processes to develop mathematical ideas through coding, and how those perceptions varied after eight weeks of professional learning.

The project was one of the most successful ones I have run: teachers loved the materials, gained confidence in teaching coding (statistically significant!) and also **learned** coding. Furthermore, their students also showed significant improvement in their learning of coding and computational thinking but also in their enjoyment of mathematics.

This research formed the pilot base of one of my PhD student's theses which proved to be a great success as signified in his appointment into the NSW Department of Education as a researcher.

Using Computer Programming to Bring Variables to Life in Grades 3-4 in Ontario, Canada

George Gadanidis, Western University

The Ontario mathematics curriculum, grades 1-9, integrates computer programming with mathematics, in algebra and also across the other strands.

The concept of variable is in the curriculum starting in grade 1

- Grade 1: identify quantities that can change and quantities that always remain the same in real-life contexts.

- Grade 2: identify when symbols are being used as variables and describe how they are being used.
- Grade 3: describe how variables are used and use them in various contexts as appropriate.
- Grade 4: identify and use symbols as variables in expressions and equations.

In several grades 3-4 classrooms, we have introduced variables through coding puzzles. This is adapted from Gadanidis, G. (2022). Coding + Mathematics: Lessons learned from research classrooms. https://learnx.ca/lessons-learned

1. Run the code below. What does it do? How does it do it?
2. Edit the code to make it draw the spiral shown on the right.

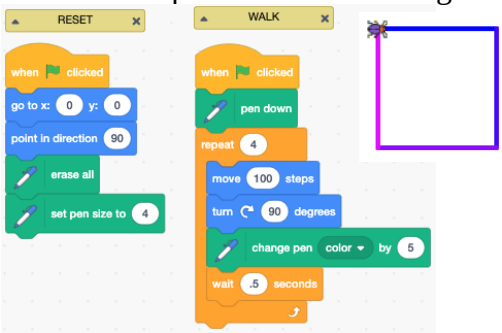

Figure 4. Code adapted from Gadanidis, G. (2022). Coding + Mathematics: Lessons learned from research classrooms (https://learnx.ca/lessons-learned), and a solution goal.

Students are able to solve the puzzle using the method of the partial code shown in figure 5 below.

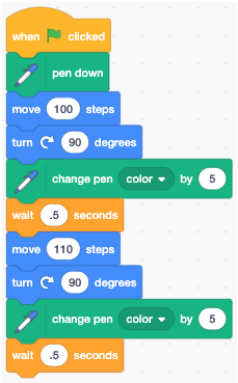

Figure 5. Students may use this method of partial code to solve the puzzle in Figure 4.

When we then ask, “Can you also solve the puzzle using a repeat block?” they are confused.

How is it possible to repeat something that needs to change?

Our approach is to then give them the code that solves this puzzle, and then present them with a new related puzzle (figure 6).

1. I found this code that seems to solve the puzzle.
2. Run the code. What does it do? How does it do it?
3. Edit the code to draw the spiral patterns shown on the right.
2. What other spiral patterns could you draw?

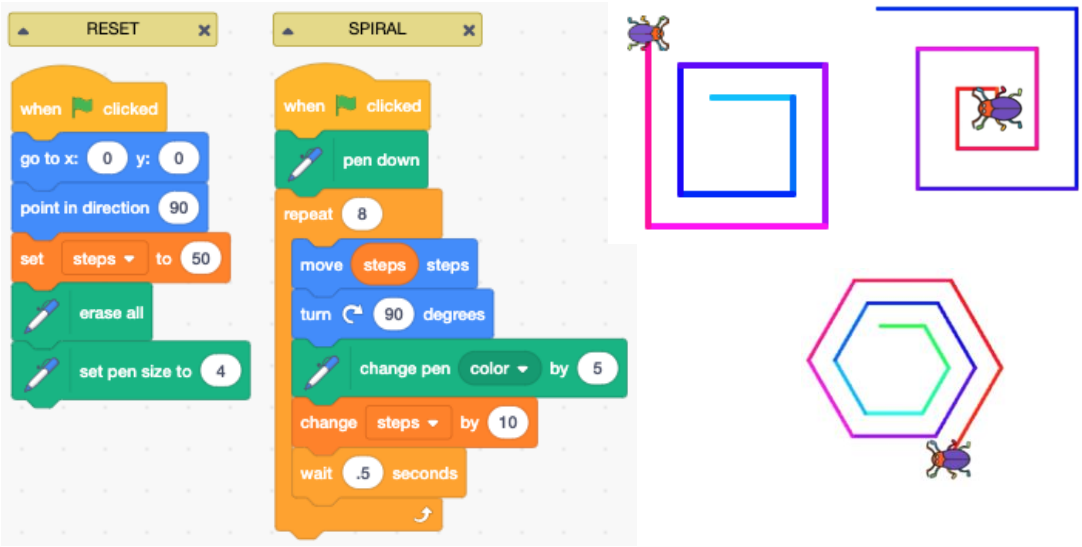

Figure 6. Code to solve previous puzzle, as well as new related puzzle and a solution goal.

Notice that we don't explicitly "teach" children how to code. They learn it incidentally as they run and edit code that works to solve related puzzles.

Coding can revitalize mathematics:

> The integration of computer programming as a modelling tool may reform and revitalize mathematics education, by remediating and reorganizing mathematical concepts and relationships and consequently changing what and how mathematics is taught, and who can learn it and when. (diSessa, 2018)

... if applied to deeper mathematics:

> Students – humans – thirst for deep connections, for such vistas, to have the space to flex their imagination, to wonder, to be surprised, to learn within the beauty that surrounds them, at least occasionally. (Gadanidis, Borba, Hughes & Lacerda, 2016)

References

ddiSessa, A. A. (2018). Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education. *Mathematical Thinking and Learning, 20*(1), 3-31.

Gadanidis, G. (2022). Coding + Mathematics: Lessons learned from research classrooms. Available at https://learnx.ca/lessons-learned

Gadanidis, G., Borba, M., Hughes, J. and Lacerda, H. (2016). Designing aesthetic experiences for young mathematicians: A model for mathematics education reform. *International Journal for Research in Mathematics Education, 6*(2), 225-244.

Reaction

Richard Noss

In his reaction, Richard noted that this had been one of the most interesting conferences of this type and the community deserved 'a pat on the back' for all the progress they had made in both theory and practice. It seemed to him that the use of computers in mathematics was no longer questioned; it is almost taken for granted so teachers can grab the autonomy that was on offer. He mentioned the proof of the four-colour theorem (Appel and Haken in 1976), which states that no more than four colours are required to colour the regions of any map so that no two adjacent regions have the same colour) and how first proof with computers provoked considerable antagonism – the computer did nothing creative, simply served as a counting

device. The presentations in the panel pointed to more creative ways for computing, programming to interact with mathematics in 'everyday' classrooms.

Summary and Reflections

Celia Hoyles

The panel exceeded my expectations, with 65 participants from 10 countries. This probably reflects the dramatic uptake of programming in Mathematics and also the increasing awareness of critically important complexities and possible inequities. I recommend readers watch the recording of the session to better capture all that went on. The recording can be found here: https://cpmath.ca/symposium-proceedings/ or directly here: https://www.youtube.com/watch?v=8YCMSvs010M&t=1s

All the participants seek equitable outcomes in practically meaningful ways, where the partnership and interactions of coding and mathematics enriches both. I pointed to the potential of a dynamic mathematics where students have autonomy over their work and its evaluation but also noted the diverse ways coding has been introduced into the school systems. In mathematics there is always resistance to change: I recalled PME 10 In London when in his keynote Seymour Papert demonstrated an elegant proof of the circle theorems. This was greeted with delight by some (Richard and me to name but two) but horror by others who rejected this disruption of the school mathematics representational infrastructure.

The panellists took different approaches to their task. On Nam presented the history of coding and then AI in the Korean school curriculum with fascinating illustrations from words textbooks along with a video of high school geometry class. She also reported notably that the Ministry provided 60 hours of teacher training for this new curriculum.

Simon presented a fascinating 'generic' example of how computer science was introduced in France referring to a framework of prescribed. Implemented and attained curriculum. He noted the heterogeneity of practice, the expertise required of the teachers to help the students and the challenge of the transition between paper and pencil and computer work,

George shared some wonderful activities devised for 'Maths with coding' where code was given to the students and they were invited to try it, puzzle over it, explain the outcomes, and make changes as they saw fit (spiralling out rather than in for example) with some multimodal activity with students acting out ideas in the classroom. He referred to coding as a potential literacy in the sense of diiSessa and his four Rs: Reformulate, Revitalise, Remediate and Reorganise. He suggested he had found this a fruitful way to analyse the potential of tasks with the proviso that the mathematics was 'deep' and meaningful.

Elena talked about the history of coding and computational thinking in Australia and in New South Wales (NSW), the province where she lives; rather surprisingly not the same in turns out. In NSW, coding appears in science rather surprisingly for us from other countries. She referred to a large teaching guide produced by the state with links to a range of illustrative activities. She herself had led a project implementing ScratchMaths (the project directed by Celia and Richard in English schools, see for example Benton, L., Kalas, I., Saunders, P., Hoyles, C., & Noss, R. (2018) Beyond Jam Sandwiches and Cups of Tea: An Exploration of Primary Pupils' Algorithm-Evaluation Strategies *J of Computer Assisted Learning* https://doi.org/10.1111/jcal.12266)

Elena reported significant improvements in coding and in computational thinking along with enhanced enjoyment in maths the classroom. However, she noted that there was a problem with teacher expertise both these areas. We were also rather shocked when she mentioned that

'everything had disappeared, the manual for example as new priorities have superseded coding …

Celia mentioned she had written a long time ago about how she had believed the computer would catalyse change but had been proved wrong – the system took over, a 'Trojan mouse' phenomenon sadly. See' 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.

A more recent example of this phenomenon is that in England where computing is compulsory once unplugged activities were recognised as important planned to complement hands-on work, unplugged activities have taken over in practice and students rarely have the chance to actually interact with computers - -it saves time and hassle for schools -- but of course misses the point.

In the final discussion, concerns were voiced about the digital divide, and I pick out a few points that I found personally of particular interest. Simon mentioned that IPADS were the preferred 'digital technology' in France and of course it is hard to code with them so the technology actually 'dictates' what happens in classrooms. The issue of gender participation in Computer Science was raised. It seems that in every country represented on the panel female participation in CS was a major concern. Elena also suggested that when computers were given to schools for a particular purpose, they in fact were rarely used for that purpose. On a more positive note, George mentioned how teaching coding in mathematics led to 'seeing something new in students'; it brought a fresh lens on to who engaged with the subject and how. This is certainly something that Richard and Celia have found over the years and regard it as a fundamentally important way to combat fixed hierarchies as to who is 'good and 'bad' at the subject (see for example Noss, R. and Hoyles, C. (1996) *Windows on Mathematical Meanings: Learning Cultures and Computers*. Dordrecht: Kluwer Academic Publishers)

Other challenges to change were mentioned such as high stakes testing, the proliferation of resources but it was reiterated that we need a *long*-term view, learning from the past while planning for the future. We in the research community can only plant seeds to stimulate and support critical learners. Let us maintain our vision of the outstandingly and unique positive possibilities of this work: the high motivation, the sense of student agency and bring the joy back to learning mathematics for all students.