UK Bratislava

UCL

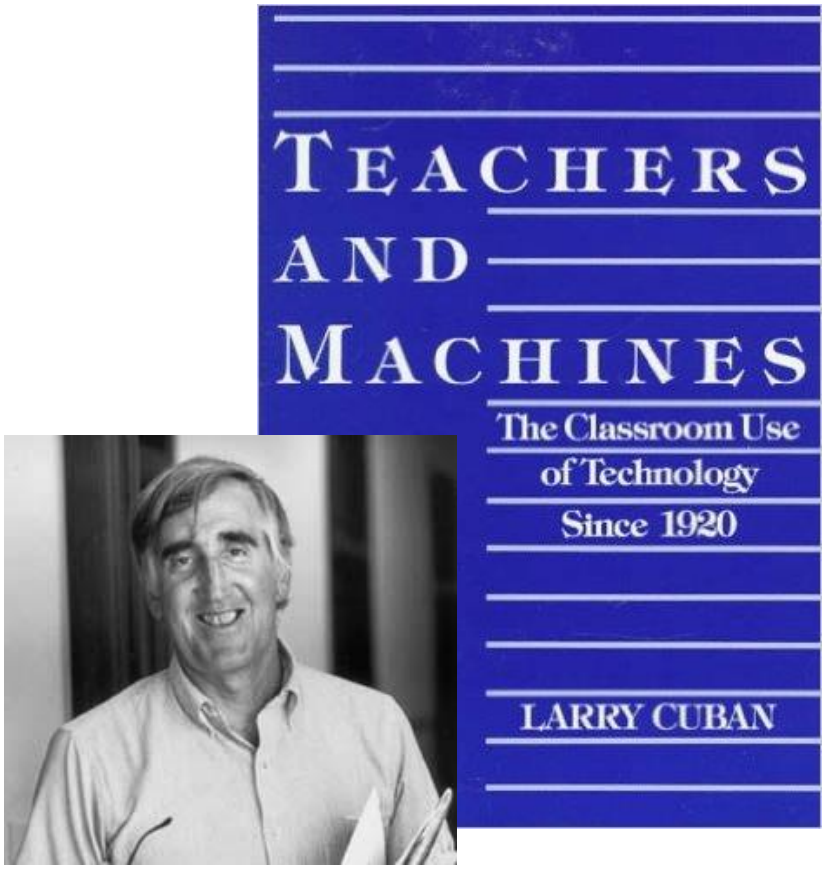
Knowledge Lab, London

Programming at School

Slovakia and Beyond: The Future

Ivan Kalas

Comenius University, Bratislava
UCL Knowledge Lab, London



1986: four waves of technologies

- big promises, big expectations
- similar obstacles…

4th wave – DT

- similar problems? **YES**
- any difference? **YES**

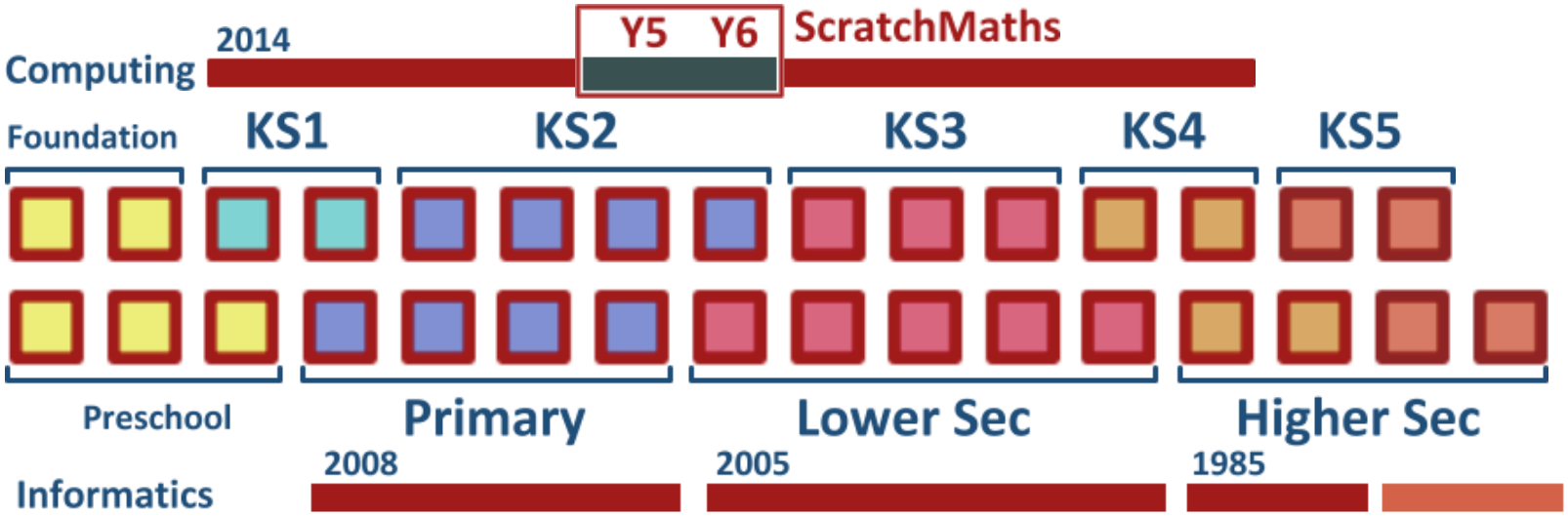
programming is its key part

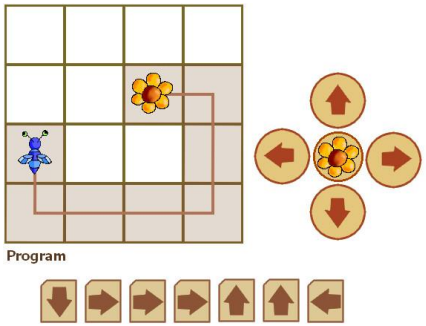
- programming at school
- educational programming
- its own waves ('sub waves')
- this one is **promising**

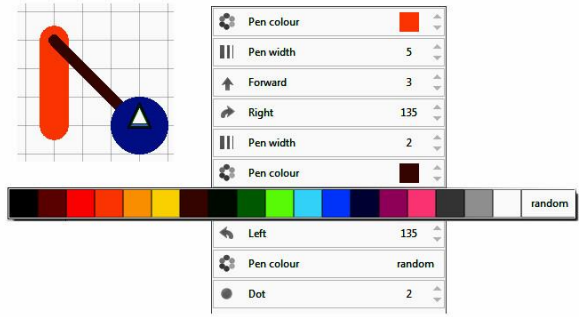
different circumstances (and goals)

- how to integrate it into general education
- what to promise, what to expect
- how to make it **sustainable and productive**

- completely new emphasis — programming for every pupil
- at all school stages — perceive it as a holistic process (but do we?)

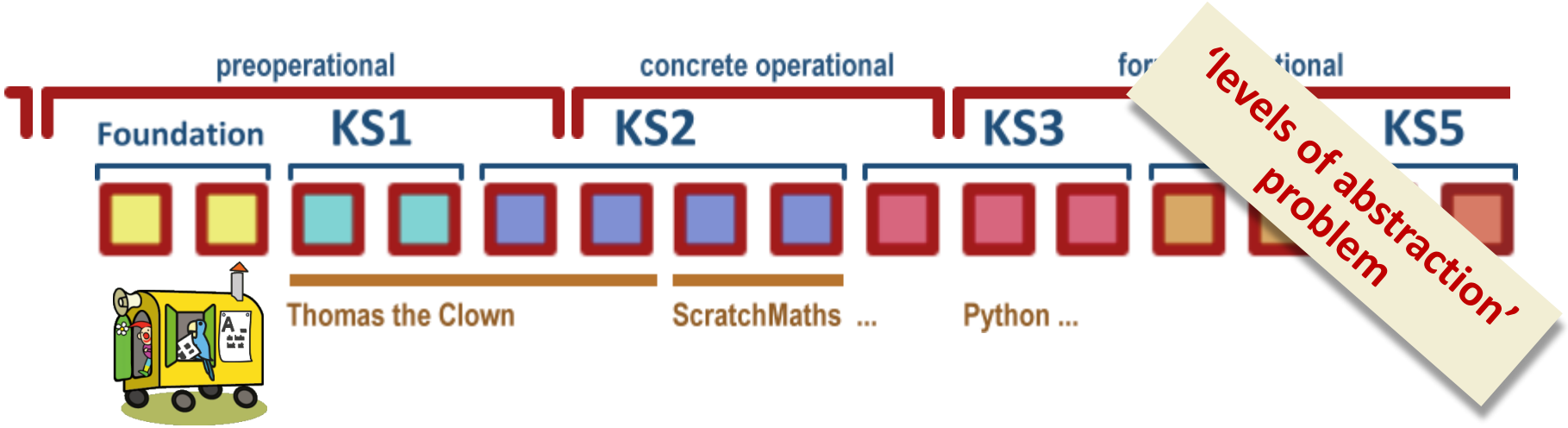






Obstacles to build programming as a holistic, well-established, and formative experience for all

- Piaget's stages of **cognitive development**



- so why should it start that early?
  what are the benefits of early educational programming?
  (new) perspective on the role of CT & programming in education

programming as a tool for learning
as an instrument for exploring, constructing

- **by integrating education programming at 11 or later**

  pupils would lose exceptional opportunities to

  create, explore, learn…
  build and experience concrete (math) concepts
  exploit powerful bridges to other areas
  cultivate problem solving skills
  develop skills like collaborating, communicating, resilience

  motivate girls… [conjecture based on Beaver]

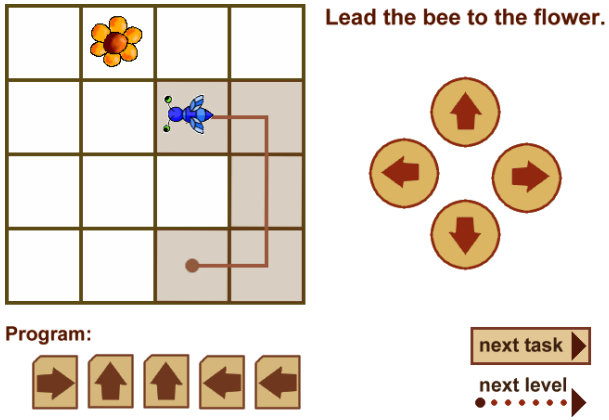
the strongest claim of my presentation

- **we would also lose the opportunity to build it as one holistic process**

  one learning process
  one understanding of its role
  but many different pedagogies – developmentally appropriate
  one consistent, supported and reflected strategy of
  transforming computational concepts from tool to tool…

Understand the process

- what are
  the concepts, related cognitive operations and their cognitive load
  unlearn to undervalue this aspect
- related computational practices
  when solving simple problem
  - in direct drive style
  - interpret given sequence
  - build a sequence for a given solution
  - read given sequence and envisage…
  - fill in 1 or 2 missing steps
  - build a sequence to solve a problem
  - build a sequence with constraints
  - select correct solution out of several
- related cognitive operations



- think of and discuss the problem
- use different strategies to solve it
- explain your solution
- record (externalise) your solution
- share a solution, learn a solution
- compare different solutions…
- …

Understand the dynamics of the process

- five stages of computing education:

  conceptual taxonomy of Fuller et al. (2007)

  interpreted from the CT perspective of Brennan and Resnick (2012)

prestructural	using unconnected concepts and practices
unistructural	local perspective of one isolated concept…
multistructural	multi-point perspective of several relevant concepts and related practices
relational	holistic perspective with meta-connections
extended abstract	generalisation, the context seen as an instance of more general case

- Lister (2016): … the stages of the novice programmers (at any age!)

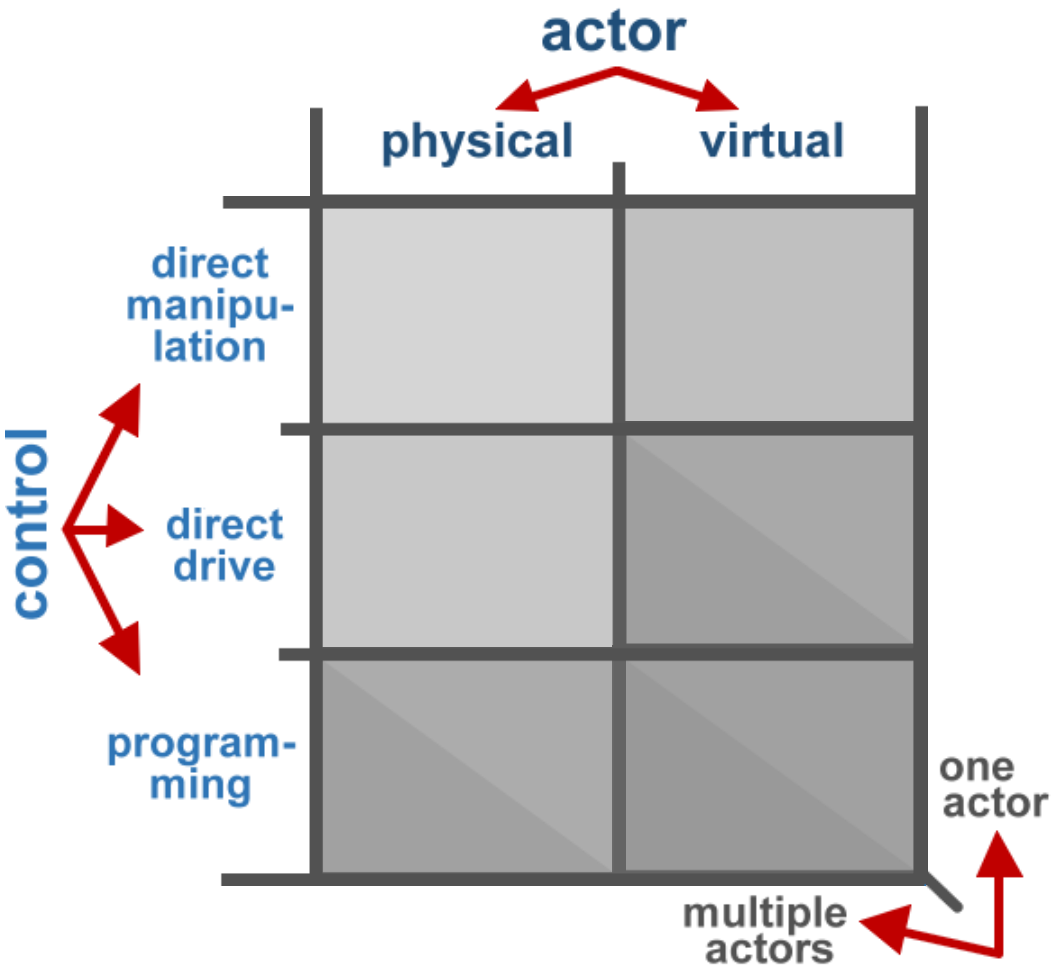
  are Piaget's stages of cognitive development (0-2, 2-7, 7-11, 11- …)

Understand the 'space of opportunities' for (primary) programming

an attempt to give it a structure – building its conceptual framework

three criteria:

- actor
- control
- interaction



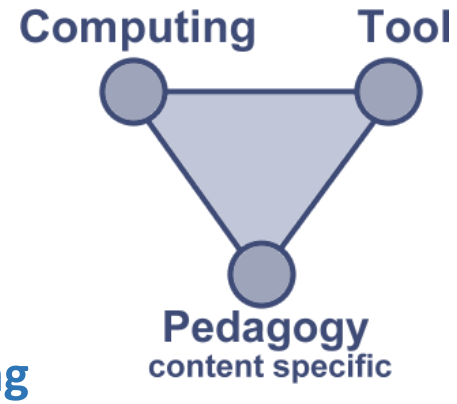
How to make this framework work for us, the developers?

- be aware of all opportunities of this space
  and exploit them
- avoid 'lethal timing' of the tool --- too early, too late, too long
  do not 'rush through' it
  --- could miss the opportunities, could destroy the process

Maintain multiple balances within content

this is **key important** for content developers!

- considering content, distinguish three aspects
  - respect Computing, CS
  - understand the Tool
  - master Pedagogy



- avoid dangerous diversions towards
  - tool
  - computing
  - pedagogy
  - technocentrism
  - CS centrism
  - shallow programming

Some more recommendations

- respect and know (primary) pupils, care about all of them
- respect (primary) teachers… they do not have any pre-service background in CS
  - develop with them *… foster a sense of teacher ownership of an innovation*
  - do not trial it for them
  - more general – build multiple ways how to get in-service qualification
- know and exploit affordances of the tool (avoid mis-affordances)
- master pedagogy
  - respect differences, encourage diversions
  - use good metaphors
  - in ScratchMaths: 5E – explore, explain, envisage, exchange, bridge
- continuously support teachers
- offer and accept bridges to all subjects
- offer inspiring, challenging and rewarding contexts – constructionism
- make it developmentally appropriate
  - which concepts, practices, tools… and when
- make it respected – by teachers, pupils, parents… and lower/higher stages
  - as an understood, productive and respectful partner in education

Thanks for your attention

Contact

kalas@fmph.uniba.sk
I.Kalas@ucl.ac.uk

References

- Brennan, Resnick (2012) New framework for studying and assessing the development of computational thinking
- Fuller et al. (2007) Developing a Computer Science-specific Learning Taxonomy
- Kabatova, Kalas, Tomcsanyiova (2016) Programming in Slovak Primary Schools
- Kalas (2016) On the Road to Sustainable Primary Programming
- Lister (2016) Toward a Developmental Epistemology of Computer Programming

ScratchMaths *Pupils projects*

- Open the **5-Challenge Project FINAL** project and run it: drag the Beetle to the left edge, click it and answer its question.

- Observe, discuss and describe all details of what is going on here.

  How many houses?
  What pen colour, pen size, pen shade?
  How big are the houses?
  How are they positioned?