Recent developments in computer science education research

Sue Sentance

King's College London & Raspberry Pi Foundation

sue@raspberrypi.org
sue.sentance@kcl.ac.uk

University of Manchester
Computer Science Department Seminar
July 23rd 2018

Today’s talk

- □ **Overview of field**
- □ **Key areas where developments are taking place**
- □ **Some King’s projects**
- □ **Programming research summary**
- □ **Questions**

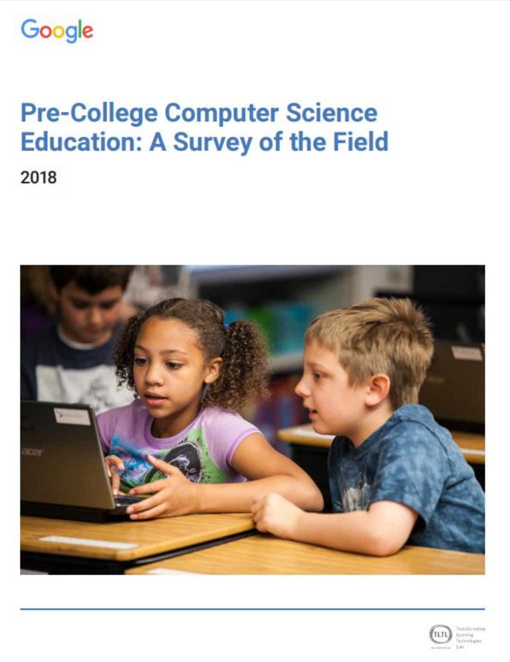
My focus is on research around computing in school but this mostly draws from CS Ed research in HE or has implications for it

↔

About me:

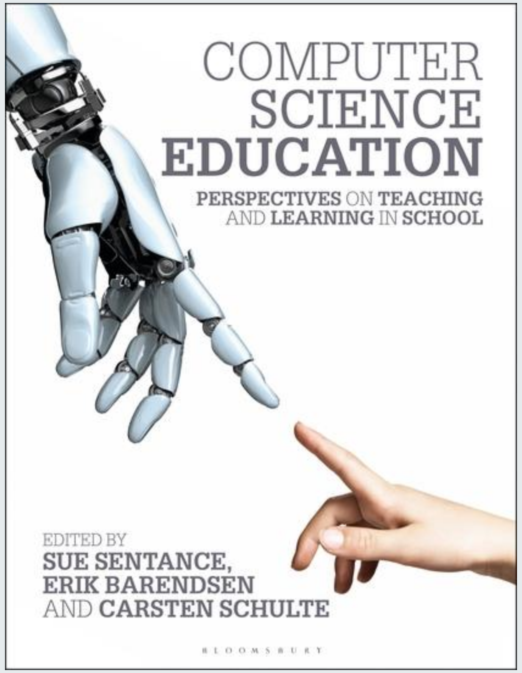
Senior Lecturer in Computer Science Education at KCL (2014-2018)
Chief Learning Officer, Raspberry Pi Foundation, 2018-..
PhD in AI & ED, 1993
PGCE 1999
Taught in schools, 1999-2010
Royal Society Advisory Committee
Computing At School/ BCS Boards etc,

Where to start?


Google

Pre-College Computer Science Education: A Survey of the Field

2018


COMPUTER SCIENCE EDUCATION

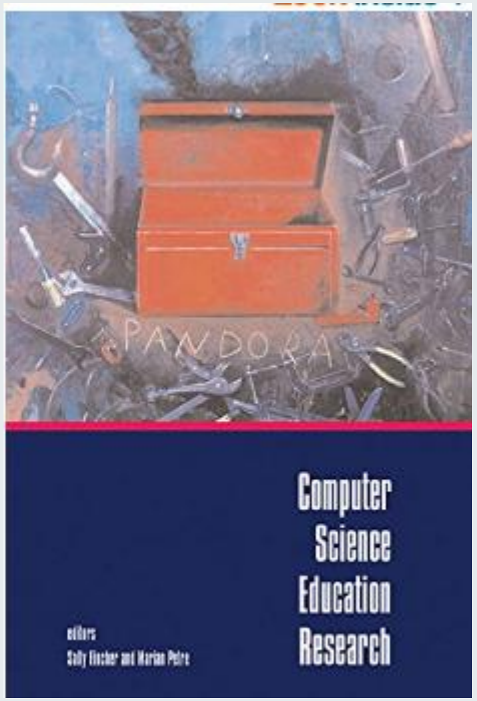
PERSPECTIVES ON TEACHING AND LEARNING IN SCHOOL

EDITED BY
SUE SENTANCE,
ERIK BARENDSEN
AND CARSTEN SCHULTE

BLOOMSBURY

After the reboot:
computing education
in UK schools

THE ROYAL SOCIETY



editors
Sally Fincher and Marian Petre

Computer Science Education Research

Journals	Conferences
Computer Science Education	ICER (more theoretical)
ACM TOCE	ITICSE (Europe)
BJET	SIGCSE (US)
Computers & Education	WIPSCE (School-focus)
Computers in Human Behaviour	ISSEP (School-focus)

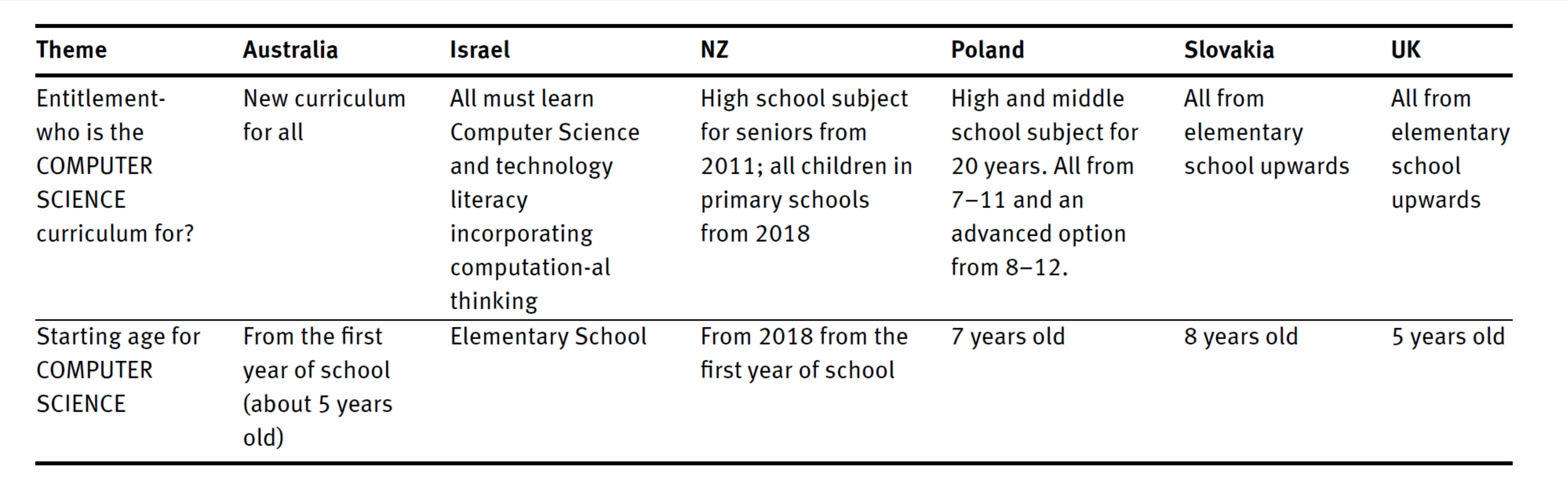
Pedagogy in teaching
Computer Science
in schools:
A Literature Review
Jane Waite, QMUL & KCL

Assessment in Computer Science courses:
A Literature Review

Maria Kallia, KCL

Our context: interest in teaching computing in school is growing

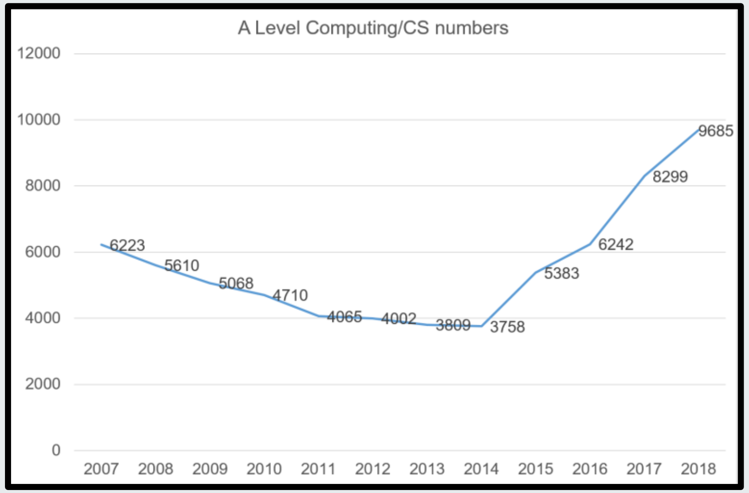
1. Internationally – a snapshot from 6 countries (Webb et al, 2018)



Theme	Australia	Israel	NZ	Poland	Slovakia	UK
Entitlement-who is the COMPUTER SCIENCE curriculum for?	New curriculum for all	All must learn Computer Science and technology literacy incorporating computation-al thinking	High school subject for seniors from 2011; all children in primary schools from 2018	High and middle school subject for 20 years. All from 7–11 and an advanced option from 8–12.	All from elementary school upwards	All from elementary school upwards
Starting age for COMPUTER SCIENCE	From the first year of school (about 5 years old)	Elementary School	From 2018 from the first year of school	7 years old	8 years old	5 years old

2. In England, Computing is a mandatory school subject

Increase in A-Level numbers



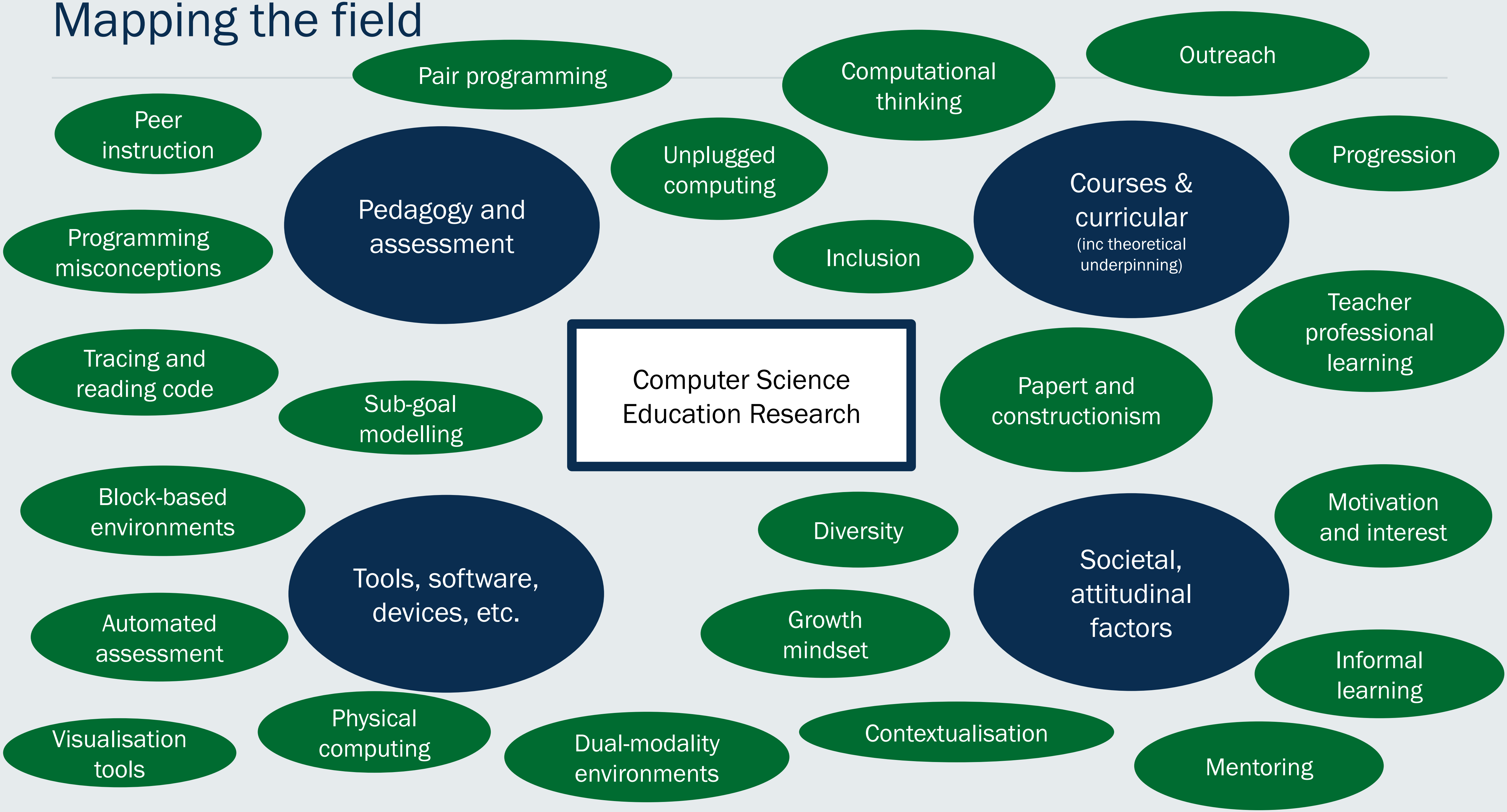
Key questions for researchers

How do we teach CS in schools and in HE?
How do we increase engagement and motivation?
.....

Key issues for researchers

In HE, CS Ed research is under-valued and under-funded
For K-12, CS Ed research is in its infancy, under-funded and not prevalent in education departments

Mapping the field



Computer Science Education Research

Pedagogy and assessment

Pair programming

Peer instruction

Programming misconceptions

Tracing and reading code

Sub-goal modelling

Unplugged computing

Computational thinking

Outreach

Progression

Courses & curricular (inc theoretical underpinning)

Inclusion

Teacher professional learning

Papert and constructionism

Block-based environments

Tools, software, devices, etc.

Automated assessment

Visualisation tools

Physical computing

Dual-modality environments

Diversity

Growth mindset

Societal, attitudinal factors

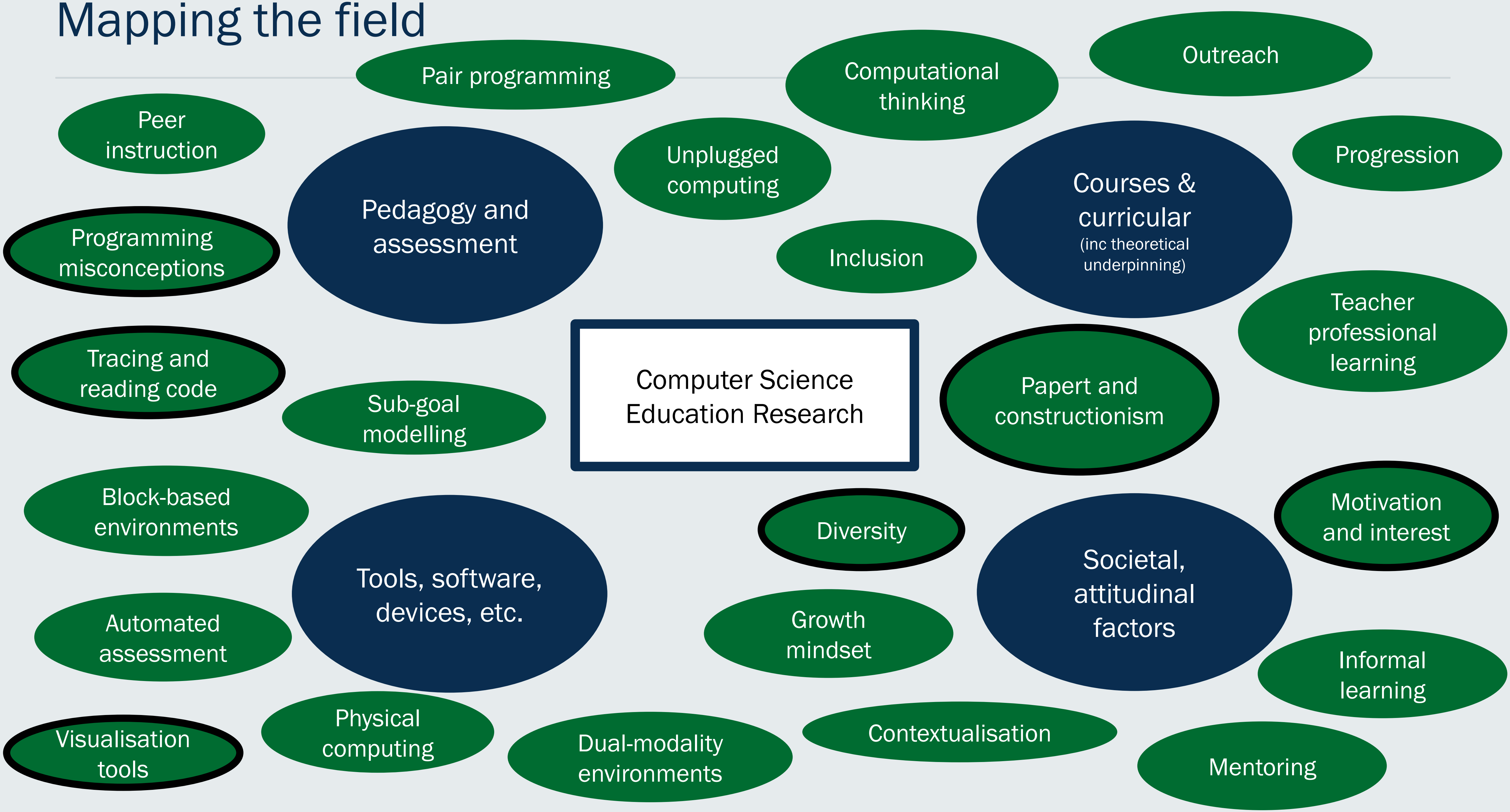
Motivation and interest

Informal learning

Contextualisation

Mentoring

Mapping the field



Computer Science Education Research

Pedagogy and assessment

Courses & curricular (inc theoretical underpinning)

Tools, software, devices, etc.

Societal, attitudinal factors

Pair programming

Computational thinking

Outreach

Peer instruction

Unplugged computing

Progression

Programming misconceptions

Inclusion

Teacher professional learning

Tracing and reading code

Sub-goal modelling

Papert and constructionism

Block-based environments

Diversity

Motivation and interest

Automated assessment

Growth mindset

Informal learning

Visualisation tools

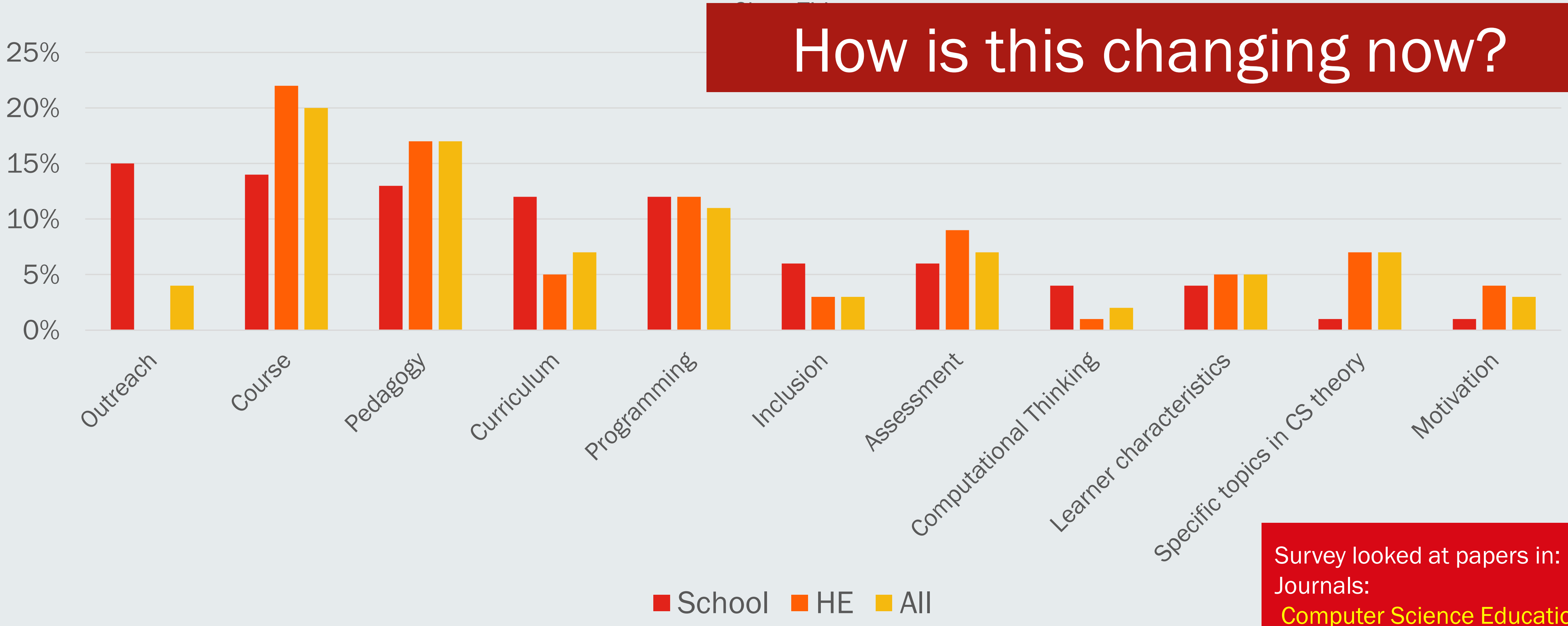
Physical computing

Dual-modality environments

Contextualisation

Mentoring

Themes from CS Ed articles 2004-2014



How is this changing now?

Numbers of papers: All – 2225 University - 1285 School -420

Survey looked at papers in:
Journals:
Computer Science Education
ACM TOCE
Conferences:
WIPSCE, ITICSE, ICER, ISSEP and SIGCSE

Some particular areas to look at

A. Programming environments for beginners

B. Learning programming: pedagogy (inc tracing, threshold concepts, PRIMM and misconceptions)

C. Computational thinking

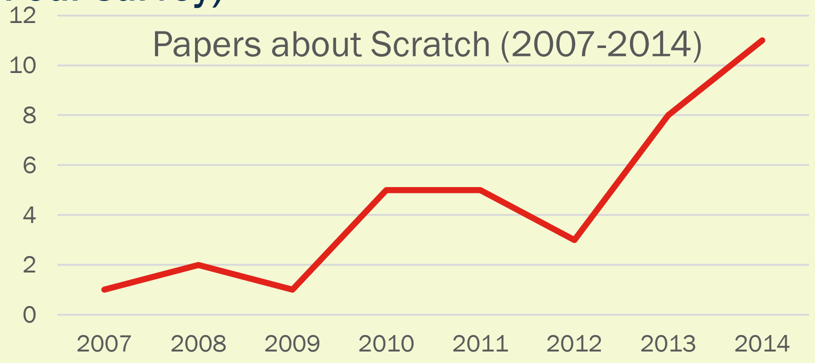
D. Physical computing

E. Inclusion

A. Programming environments

A. Programming environments for beginners

Increase in numbers of papers about Scratch since 2007 (in our survey)



Some key papers

Learning computer science concepts with Scratch (Meerbaum et al, 2013)

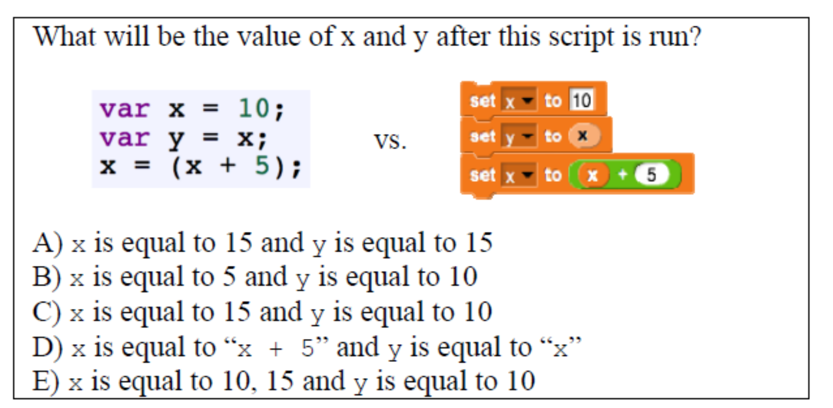
- Focus on concepts learned (not skills)
- Tested > 200 Year 9 students taught Scratch systematically
- Testing showed difficulties with loops, variables and concurrency

Habits of programming Scratch (Meerbaum-Salant et al, 2011)

- block-based environments increase extremely fine-grained programming (too bottom-up) – bad habits for future

Recent developments

Recent developments: comparing understanding in blocks-based and text-based programs (Weintrop and Wilensky, 2015)



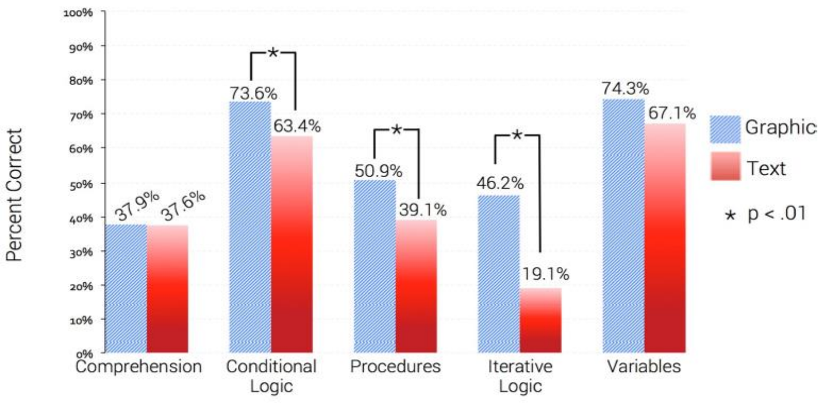
What will be the value of x and y after this script is run?

```
var x = 10;
var y = x;
x = (x + 5);
```

vs.

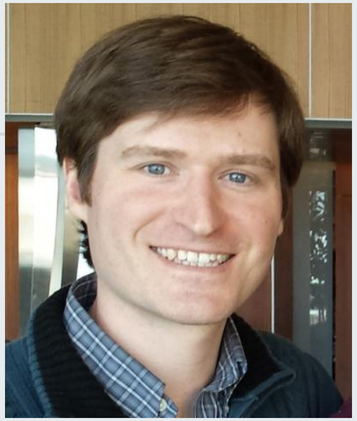
```
set x to 10
set y to x
set x to x + 5
```

A) x is equal to 15 and y is equal to 15
B) x is equal to 5 and y is equal to 10
C) x is equal to 15 and y is equal to 10
D) x is equal to "x + 5" and y is equal to "x"
E) x is equal to 10, 15 and y is equal to 10



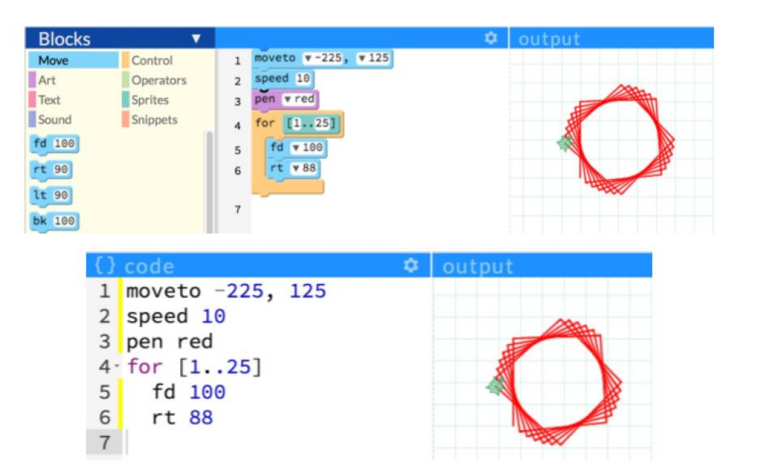
Another paper last year found that although attitudes and perceived difficulty was the same with block-based and text-based programming, pupils achieved goals more quickly (less idle time) with block-based (Price & Barnes, 2015).

A: Dual-modality programming environments



David Weintrop and Nathan Holbert, 2017

From Blocks to Text and Back: Programming Patterns in a Dual-modality Environment



```
moveto -225, 125
speed 10
pen red
for [1..25]
  fd 100
  rt 88
```

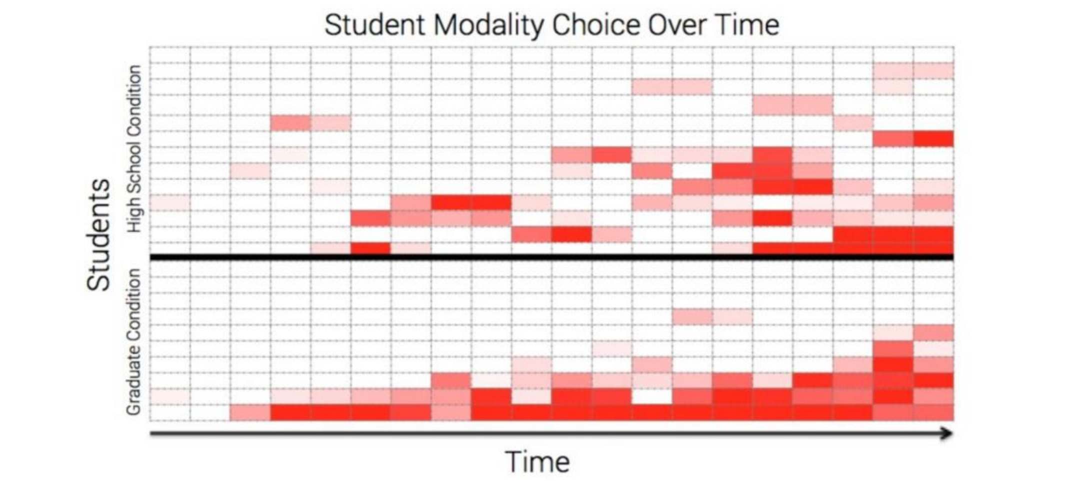


Figure 3. Student modality choice over time – the darker the square, the more time spent in the text interface.

23 students – 13 high school, 10 in HE
Initially all students worked in blocks
Students switch modes – not one-way
Correlation between self-efficacy and use of text-based

Frame-based editing with Stride

A different approach is being taken by the Greenfoot team at KCL (Michael Kölling, Neil Brown et al) who use frame-based editing as a way of programming using text but a drag-and-drop highly structured interface

B. Learning programming

B. Learning programming : reading code

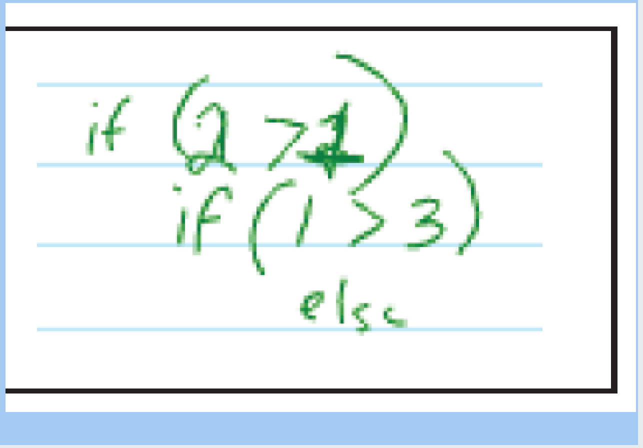
Some research highlights

2004: Multi-institutional study of reading and tracing skills shows better performance in programming by those able to trace code (Lister et al, 2004)

2011: Use of neo-Piagetian framework to establish stages that novice programmers go through (Lister et al, 2011)

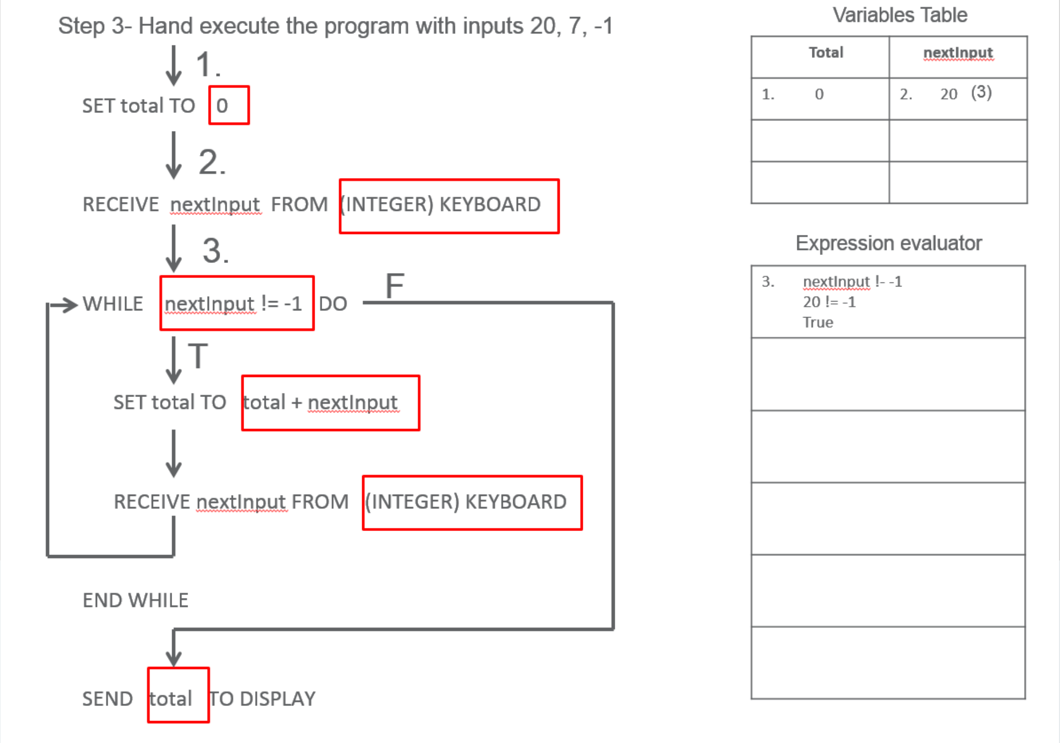
2014: Exemplification of framework through case studies, using think-aloud to find out more about students' thought process while programming (Teague and Lister,, 2014).

Work in this area suggests that student pass through neo-Piagetian stages: sensorimotor, preoperational, and concrete operational stages, before eventually reaching programming competence at the formal operational stage



Application in school

PLAN C (CAS Scotland) have developed a model for enabling students to trace through code called TRACS



Step 3- Hand execute the program with inputs 20, 7, -1

```
1.
SET total TO 0
2.
RECEIVE nextInput FROM (INTEGER) KEYBOARD
3.
WHILE nextInput != -1 DO        F
  T
    SET total TO total + nextInput
    RECEIVE nextInput FROM (INTEGER) KEYBOARD
END WHILE
SEND total TO DISPLAY
```

Variables Table

Total	nextInput
1. 0	2. 20 (3)

Expression evaluator

3. nextInput !- -1 20 != -1 True

B. Learning programming: Common misconceptions

The 1980s

Early work on misconceptions and novice programmers (Ben du Boulay, 1986, Bayman and Mayer, 1983, Bonar and Solway, 1985)

- **Loops are difficult**
- **Differentiating between a string and a number**
- **Confusion of equality and assignment**
- **Inputting data (from where?)**
- **etc…**

Steady stream of work in this field, including Juha Sorva's PhD work

One recent paper: Exploring programming misconceptions (Sirkia and Sorva, 2014)

Based on analysis of 24,000 log files from UG students first learning programming:

Research found:

- **Inverted assignment (first = second) - wrong way round**
- **If X == Y: (execute *then* part even if false)**
- **Returning False from function even though condition does not hold**
- **Not storing return value of function**
- **Etc….**

New chapter by Sorva (2018) describes 41 different programming misconceptions (in my book)

B: Examples of current projects: threshold concepts

Maria Kallia is working on a project to understand which concepts in computer programming are particularly difficult and could be identified as “threshold concepts”.

- Focus on functions and parameters in programming
- Latest study looked at liminal space – the confused state you are in before you reach an understanding of something (go over the threshold)
- Maria developed a test of programming performance and compared to attitudinal factors including self-efficacy, motivation and interest

Findings (all statistically significant):

- There is a significant relationship between liminal space and CS identity
- Troublesome knowledge impacts sense of belonging and motivation, but not identity
- Girls in the liminal group experience troublesome knowledge more intensely than boys and this influences their sense of belonging, motivation and identity while boys experience an impact only in the sense of belonging.
- She has developed a model for predicting students are in liminal or post-liminal space from their self-efficacy, motivation, identity and self-evaluation which explains 78.6% of the variance.

B: Examples of current projects: PRIMM

A framework for working with beginners using text-based programming:

Predict – given a working program, what do you think it will do? (function)
Run – run it and test your prediction
Investigate – get into the nitty gritty. What does each line of code mean? (structure). Lots of activities here: trace, annotate, explain, talk about, identify parts, etc….
Modify – edit the program to make it do different things (function)
Make – design a new program that uses the same nitty gritty but that solves a new problem (function).

PRIMM Study (to appear!)
Mixed-methods study
Part A: Quasi-experimental design
14 KS3 teachers
180 students in control group (11-13)
493 students in experimental group (11-13)
Pre-test and post-test
Findings:

- Statistically significant difference between post-tests of control and experimental groups

Part B: Co-generation, design-based research with teachers
10 interviews, 1 focus group, and teacher journals
Understanding why the approach works
Themes emerged around:

- Differentiation and increased accessibility
- Influence of language and talk

Theoretical framework

- Drawing on Vygotsky and social constructivism
- Concept of mediation is used to explain how the program begins on the social plane then moves to cognitive plane and to being understood by the learner
- This is similar to the Use-Modify-Create approach it builds on

PRIMM Materials
The materials we used in our study will shortly be available online with lesson plans for adaption by teachers

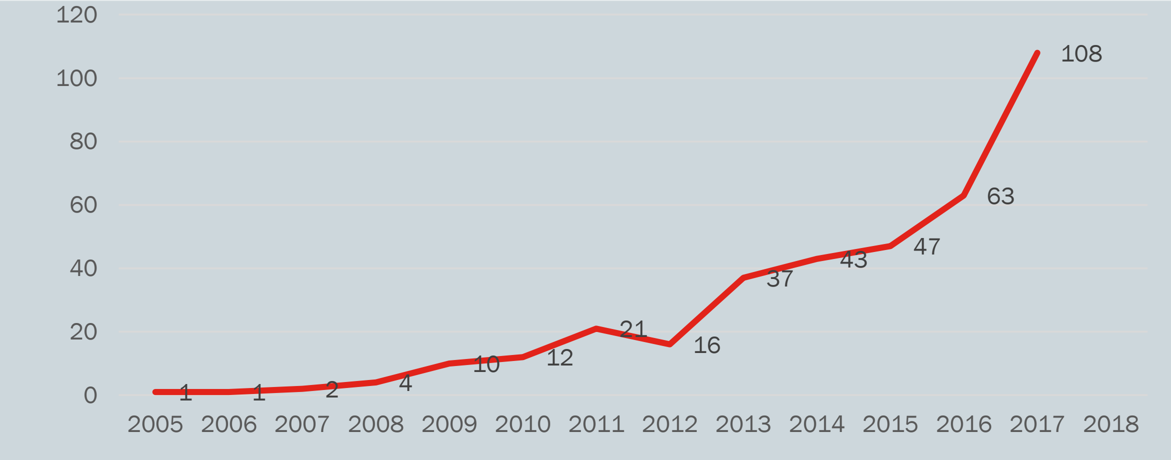
C. Computational Thinking

C. Computational thinking

CT has become a popular research topic

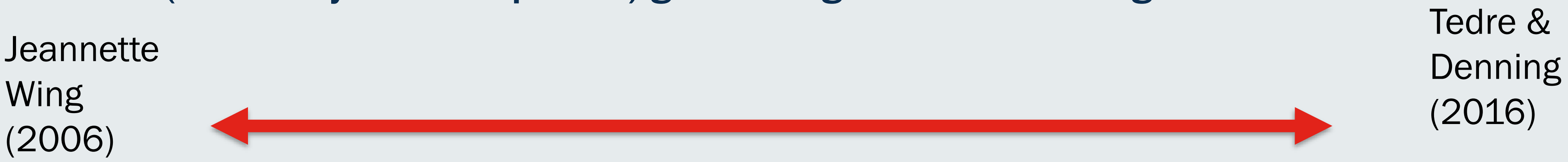
Graph shows number of papers published with CT in title

44 already in 2018



Data from https://csedresearch.wordpress.com/computational-thinking/

Different views (this analysis is simplistic!) generating some interesting debates



Jeannette Wing (2006)

Tedre & Denning (2016)

- CT is a fundamental skill
- For everybody, not just computer scientists
- Human thinking, not just computer thinking
- Ideas and problem solving

“[Computational thinking] … represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.”

- History of algorithmic thinking – we’ve been here before
- CT is not a superior way of thinking
- We should emphasise **design** and **modelling**
- Risk of exaggerated claims
- Risk of narrowing view of computing

”The new CT movement aimed to include also those who use computational tools and those who engage in step-by-step procedures. The attempt to broaden the CT audience moved into unchartered territory, where there is less certainty that tool users and procedure followers need CT or benefit from it.”

Some reviews of CT literature

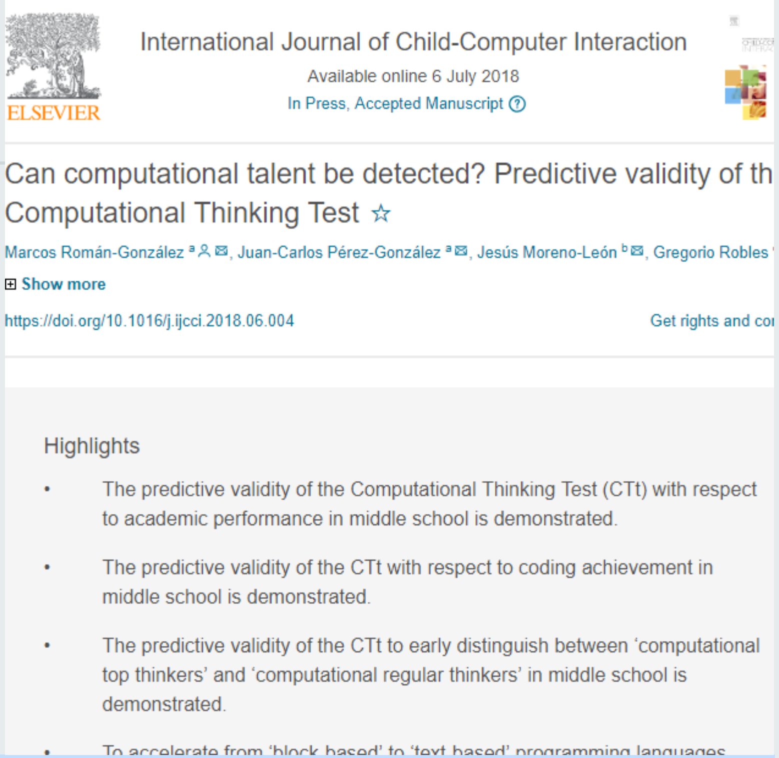
Author	Number of papers	Conclusion
Kalelioglu, Gulbahar and Kukul (2016)	125 papers on CT	Lack of theoretical framework – game-based learning and constructivism primarily Immature field – not many papers No consistent definition
Lye and Koh (2014)	27 intervention studies	Focusing on CT and programming only Need to explore more classroom-based interventions
Shute, Sun and Asbell-Clarke (2017)	45 papers reviewed	Considered papers researching CT in robotics, game design and range of environments Comparison of frameworks and proposed new one

C: Example projects in the area of CT

Recent work by a group of researchers in Madrid:

- Can computational talent be detected? Predictive validity of the Computational Thinking Test
- Extending the nomological network of computational thinking with non-cognitive factors
- Towards Data-Driven Learning Paths to Develop Computational Thinking with Scratch

Possibly the most rigorous work being conducted in the area of computational thinking



ELSEVIER

International Journal of Child-Computer Interaction

Available online 6 July 2018

In Press, Accepted Manuscript ⓘ

Can computational talent be detected? Predictive validity of th
Computational Thinking Test ☆

Marcos Román-González [a], Juan-Carlos Pérez-González [a], Jesús Moreno-León [b], Gregorio Robles

⊞ Show more

https://doi.org/10.1016/j.ijcci.2018.06.004

Get rights and co

Highlights

- The predictive validity of the Computational Thinking Test (CTt) with respect to academic performance in middle school is demonstrated.
- The predictive validity of the CTt with respect to coding achievement in middle school is demonstrated.
- The predictive validity of the CTt to early distinguish between ‘computational top thinkers’ and ‘computational regular thinkers’ in middle school is demonstrated.
- To accelerate from ‘block based’ to ‘text based’ programming languages

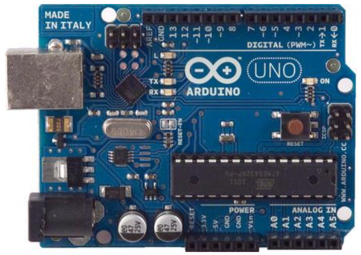
Recent results:

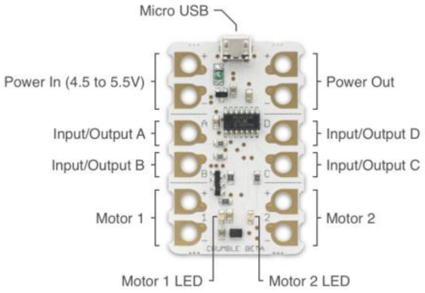
- Development of a model to predict computational talent in school students
- Evaluated with 314 middle school students
- Distinguishes between computational regular thinkers and computational top thinkers
- Implications for computing curriculum development

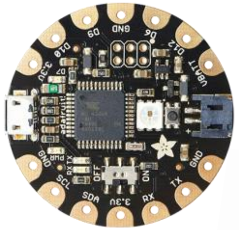
NB Their definition of CT is more like programming (warning to Wing-lovers)

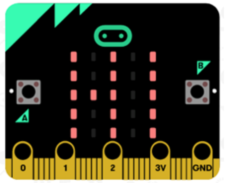
D. Physical computing

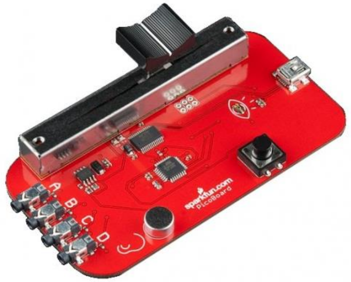
D. Physical computing: more and more devices

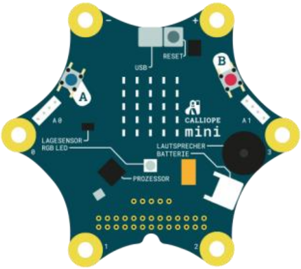










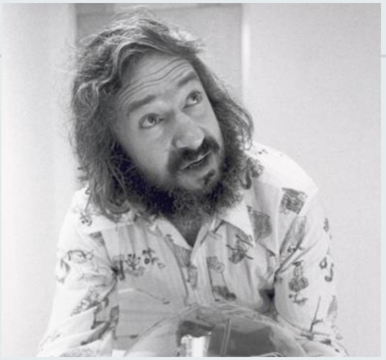


■ ■ ■ and the rest

D. Physical computing

Closely linked to constructionism in the literature:

"Constructionism –the N word as opposed to the V word – shares constructivism's connotation of learning as "building knowledge structures" irrespective of the circumstances of the learning. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on a beach or a theory of the universe." (Papert, 1991)



Research on pedagogical benefits

Tangible nature may facilitate understanding (Marshall, 2007)

Concrete easier to understand than abstract (Papert, 1980)

Lends itself to collaborative working (Sentance & Schwiderski-Grosche, 2013)

Can learn directly about how computers work and mathematics (Papert, 1980)

Facilitates creativity (Kafai, 2015)

Hybrid interfaces can be used to facilitate progression in programming (Horn et al. 2012).

Examples of pedagogical approaches:

Activity-media design (Jin et al, 2016)
Method developed to facilitate the development of physical computing learning activities which minimise cognitive load.

Use-modify-create (Lee et al 2011)
Move from "not mine" to "mine"
Use existing projects first, then modify and build new ones

E. Inclusion

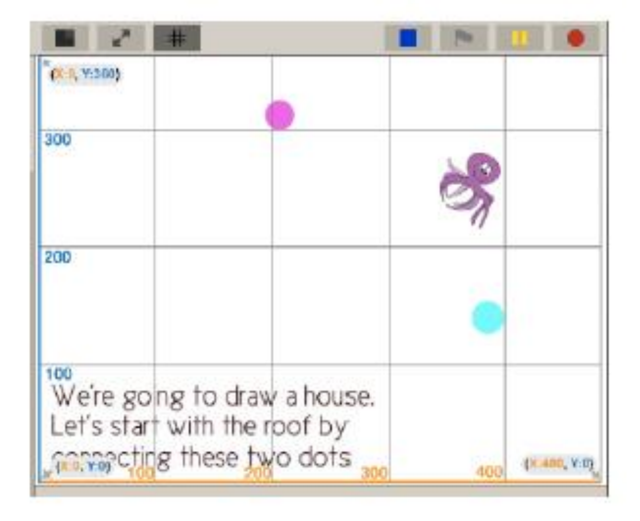
E. Inclusion

Key question:

How can we build a Computing curriculum that is accessible to all learners?

Hansen et al, 2016. Differentiating for Diversity: Using Universal Design for Learning in Elementary Computer Science Education.

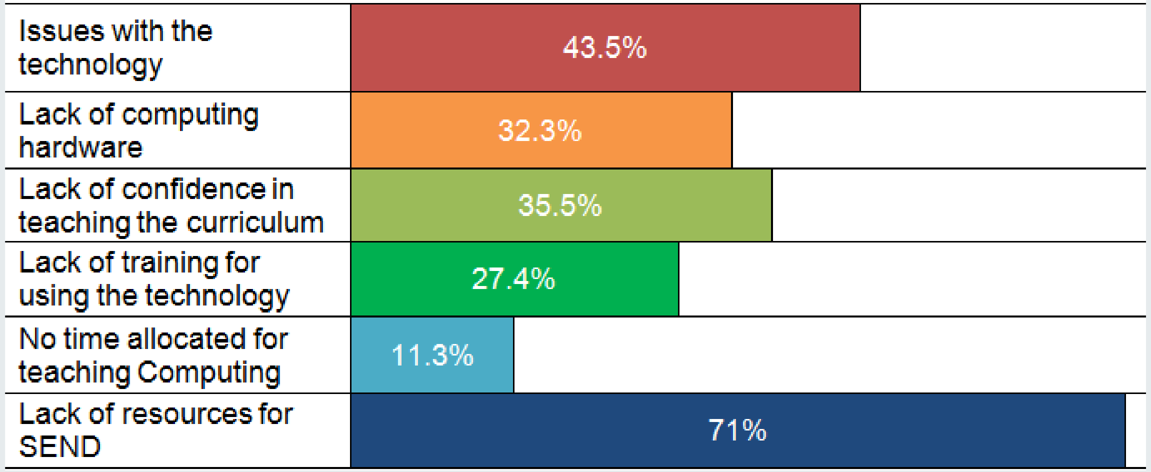
Design-based research methodology used to iteratively inform the development of the curriculum, programming environment, and research - involves researchers and practitioners collaborating in real-world settings with the aim of improving educational practices.



We're going to draw a house. Let's start with the roof by connecting these two dots

Catherine Elliott, presenting at LCERS

65 special-needs teachers reported on opportunities and barriers of computing in the SEND classroom



“Many of our students are very engaged by ICT. It can be incredibly motivational for ASD and SEMH learners. It empowers students who struggle with social interaction to present and share their work widely.”

Need for more research!

E: Examples of current projects: Torino and visually impaired children

Torino is a physical programming language that was developed to be inclusive of learners with visual impairments. It was designed to teach programming concepts to children ages 7-11 regardless of level of vision.

To create programs with Torino, physical ‘command pods’ are connected together, which produce sound in the form of music, stories and poems. There are four main types of command pods: play, pause, loop and selection, each of which represents a line of code in the program.

Alex Hadwen-Bennett is looking at the way that visually-impaired learners use Torino to learn programming

Initial findings:

- Blind and partially-sighted students use the tool in different ways to learn programming
- Exploratory procedures (type of gestures) are used to trace the flow of control in the program
- Different types of gestures and exploratory procedures are used to demonstrate understanding

Potential implications for sighted students:

The focus on control flow as a separate process to tracing is being explored through this work -> implications for work in schools on teaching programming.

Programming research summary

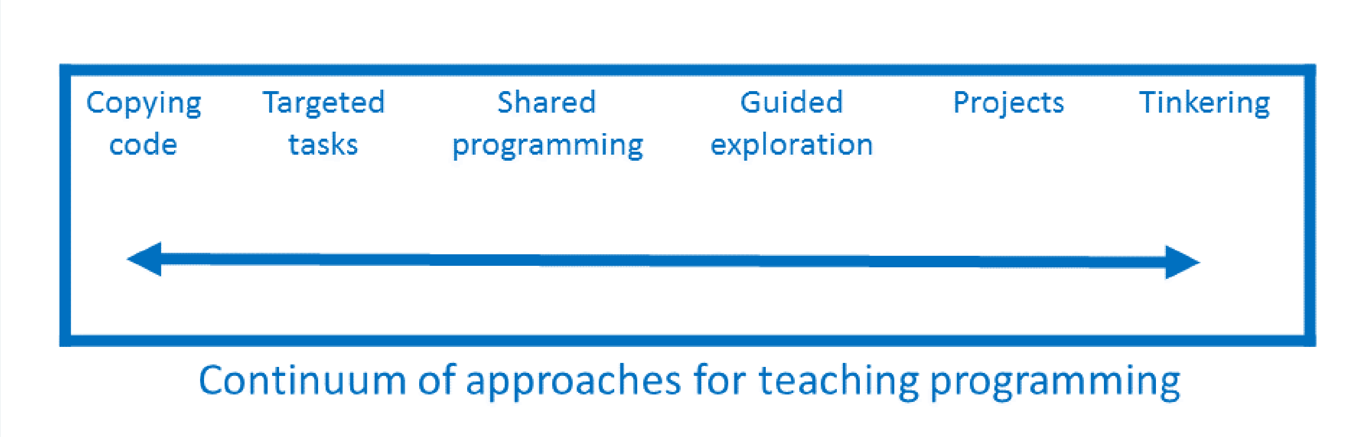
We have already seen some research in the area of learning programming:

- Effectiveness of reading code
- PRIMM
- Threshold concepts
- Block-based and dual-modality programming environments

Other research (with no time to cover) includes:

- Stepwise self-explanation
- Sub-goal modelling
- Worked examples
- Pair programming
- Use of Parson’s Puzzles
- Tinkering and Bricolage

See Caspersen (2018) for an overview



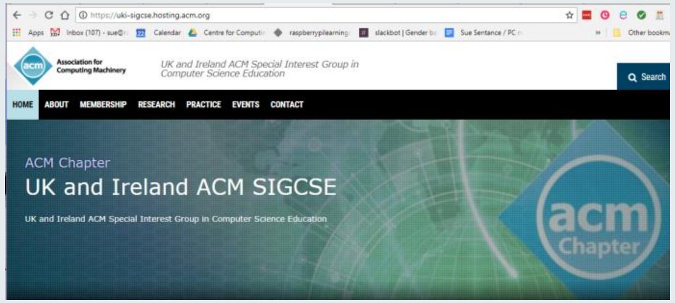
Continuum of approaches for school education by Jane Waite (https://blogs.kcl.ac.uk/cser/2018/01/05/a-continuum-of-scaffolding/)

What next?

Upcoming meetings and conferences around computing education research

Event	Date
ICER 2018	Aug 13-15, Helsinki
WIPSCE 2018	Oct 4-6, Potsdam
ISSEP 2018	Oct 10-12, St Petersburg
CAS Research meetings	October half-term, Feb half-term, etc.
Computing Education Practice conference	Jan 9, Durham
SIGCSE 2019	Feb 27-Mar 2, Minneapolis
LCERS at King's College London	June
ITICSE 2019	July 15-17, Aberdeen

Find this and more information at the UK -ACM SIGCSE website: https://uki-sigcse.hosting.acm.org/



UK and Ireland ACM Special Interest Group in Computer Science Education

ACM Chapter

UK and Ireland ACM SIGCSE

UK and Ireland ACM Special Interest Group in Computer Science Education

Take-away thought

This should be an exciting time to be involved in computing education research due to the emergence of the subject in schools. The opportunities provided by new computing curricula coupled with advances in technologies and analytical tools with which to mine big datasets, and the increasingly interdisciplinary nature of educational research, offer enormous scope for advancing computing teaching and learning.

The Royal Society’s After the Reboot report, 2018

References

Bayman, P., & Mayer, R. E. (1983). A diagnosis of beginning programmers' misconceptions of BASIC programming statements. *Communications of the ACM*, *26*(9), 677-679.

Boulay, B. D. (1986). "Some Difficulties of Learning to Program." Journal of Educational Computing Research 2(1): 57-73.

Horn, M. S., Crouser, R. J., & Bers, M. U. (2012). Tangible interaction and learning: the case for a hybrid approach. Personal and Ubiquitous Computing, 16(4), 379–389.

Jin, K., Haynie, K., and Kearns, G. (2016). Teaching Elementary Students Programming in a Physical Computing Classroom. Proceedings of the 17th Annual Conference on Information Technology Education. Boston.

Kafai, Y. B. and Burke, Q. 2013. The social turn in K-12 programming: moving from computational thinking to computational participation. In *Proceeding of the 44th ACM technical symposium on Computer science education* (SIGCSE '13). ACM, New York, NY, USA, 603-608

Kalelioglu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic Journal of Modern Computing*, *4*(3), 583.

Kallia, M. (2017). Assessment of computer science courses: a literature review. Part of the Royal Society Computing Education project. Can be accessed at: https://royalsociety.org/topics-policy/projects/computing-education/

Lister, R. (2011, January). Concrete and other neo-Piagetian forms of reasoning in the novice programmer. In *Proceedings of the Thirteenth Australasian Computing Education Conference-Volume 114* (pp. 9-18). Australian Computer Society, Inc..

Lister, Raymond, Elizabeth S. Adams, Sue Fitzgerald, William Fone, John Hamer, Morten Lindholm, Robert McCartney et al. "A multi-national study of reading and tracing skills in novice programmers." In *ACM SIGCSE Bulletin*, vol. 36, no. 4, pp. 119-150. ACM, 2004.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?. *Computers in Human Behavior*, *41*, 51-61.

Marshall, P. (2007). Do tangible interfaces enhance learning? in Proceedings of TEI'07, 15-17 Feb 2007, Baton Rouge, LA, USA.

Meerbaum-Salant O., Armoni M. & Ben-Ari M., 2011. Habits of programming in scratch. In Proceedings of the 16th annual joint conference on Innovation and technology in computer science education - ITiCSE ’11. ACM Press.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with Scratch. *Computer Science Education*, *23*(3), 239-264.

Papert, S. 1980. *Mindstorms: children, computers and powerful ideas,* Brighton, Harvester. (book)

Price, T. W., & Barnes, T. (2015, July). Comparing Textual and Block Interfaces in a Novice Programming Environment. In *Proceedings of the eleventh annual International Conference on International Computing Education Research* (pp. 91-99). ACM.

Román-González, M., Pérez-González, J.-C., Moreno-León, J., Robles, G. Can computational talent be detected? Predictive validity of the Computational Thinking Test, International Journal of Child-Computer Interaction (2018), https://doi.org/10.1016/j.ijcci.2018.06.004

Sentance, S. and Schwiderski-Grosche, S. 2012. Challenge and creativity: using .NET Gadgeteer in schools. In Proceedings of the 7th Workshop in Primary and Secondary Computing Education (Hamburg, Germany, November 08 - 09, 2012). WiPSCE '12. ACM, New York

Sentance, S., & Waite, J. (2017). PRIMM: Exploring pedagogical approaches for teaching text-based programming in school. In *Proceedings of the 12th Workshop in Primary and Secondary Computing Education: WIPSCE '17.* Nijmegen.

Sentance, S., Barendsen, E. and Schulte, C. (Eds). *Computer Science Education: Perspectives on teaching and learning in school*. Bloomsbury Academic.

Sentance, S., Selby, C and Kallia, M. (2018). Assessment in the computing classroom. In Sentance, S., Barendsen, E. and Schulte, C. (Eds). *Computer Science Education: Perspectives on teaching and learning in school*. Bloomsbury Academic.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*.

Sirkiä, T., & Sorva, J. (2012, November). Exploring programming misconceptions: an analysis of student mistakes in visual program simulation exercises. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research* (pp. 19-28). ACM.

Sorva, J. (2018). Misconceptions and the Beginner Programmer. In Sentance, S., Barendsen, E. and Schulte, C. (Eds). *Computer Science Education: Perspectives on teaching and learning in school*. Bloomsbury Academic.

Teague, D., & Lister, R. (2014, June). Blinded by their Plight: Tracing and the Preoperational Programmer. In *Proceedings of the Psychology of Programming Interest Group Annual Conference 2014* (pp. 53-64).

Tedre, M. & Denning, P. J. (2016) The Long Quest for Computational Thinking. Proceedings of the 16th Koli Calling Conference on Computing Education Research , November 24-27, 2016, Koli, Finland: pp. 120-129.

The Royal Society, (2017). After the reboot: Computing Education in UK Schools. : https://royalsociety.org/topics-policy/projects/computing-education/

Waite, J. (2017). Pedagogy and computing: a literature review. Part of the Royal Society Computing Education project. Can be accessed at: https://royalsociety.org/topics-policy/projects/computing-education/

Webb, M., Bell, T., Davis, N., et al. (2018). Tensions in specifying computing curricula for K-12: Towards a principled approach for objectives. *it - Information Technology*, 60(2), pp. 59-68. Retrieved 22 Jul. 2018, from doi:10.1515/itit-2017-0017

Weintrop, D., & Wilensky, U. (2015, July). Using commutative assessments to compare conceptual understanding in blocks-based and text-based programs. In *Proceedings of the Eleventh Annual International Conference on International Computing Education*

Wing, J. M. 2006. Computational Thinking. *Communications of the ACM,* 49, 33-35.