Computing Requirements at KS1

- understand what algorithms are; how they are implemented as programs on digital devices; and that programs execute by following precise and unambiguous instructions
- create and debug simple programs
- use logical reasoning to predict the behaviour of simple programs
- use technology purposefully to create, organise, store, manipulate and retrieve digital content
- recognise common uses of information technology beyond school
- use technology safely and respectfully, keeping personal information private; identify where to go for help and support when they have concerns about content or contact on the internet or other online technologies.

Computing Requirements at KS2

- design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts
- use sequence, selection, and repetition in programs; work with variables and various forms of input and output
- use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs
- *understand computer networks including the internet; how they can provide multiple services, such as the world wide web; and the opportunities they offer for communication and collaboration*
- *use search technologies effectively, appreciate how results are selected and ranked, and be discerning in evaluating digital content*
- *select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information*
- *use technology safely, respectfully and responsibly; recognise acceptable/unacceptable behaviour; identify a range of ways to report concerns about content and contact.*

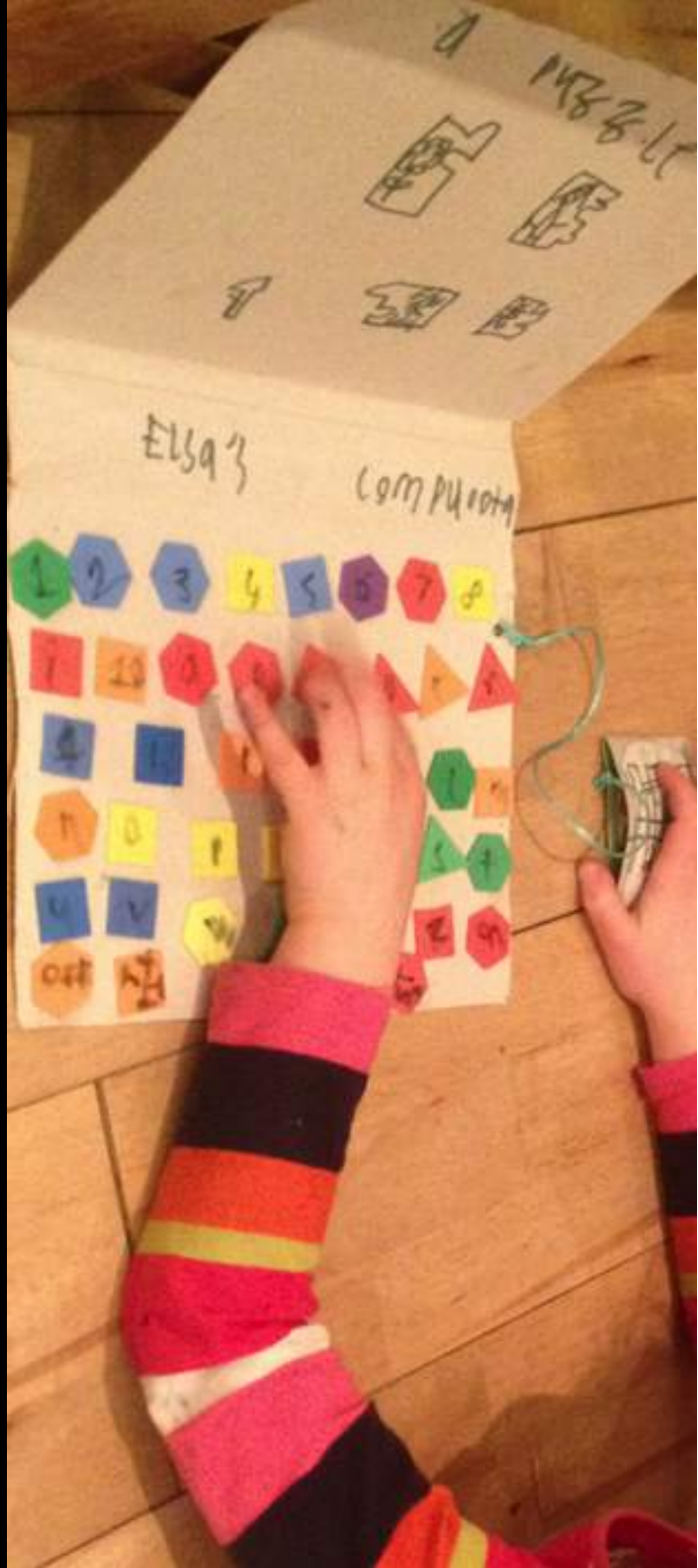
NB at KS3 NC includes "use 2 or more programming languages, at least one of which is textual"

Teaching Coding

- Autumn 2013 teaching in Glebe Primary
- Making apps/coding with kids
- Teaching elements of HTML/CSS/JS
- It was complex!

Getting Less Technical

- Concepts not syntax
- Offline
- Defining problem clearly
- Explain solution verbally (The A word)
- Progression of skills
- Publishing apps == audience



Sharing the Approach

- Structure
- Focussed projects & lessons
- Consolidation
- Tutorial videos
- Became Espresso Coding

Teaching Coding Offline

- Plan/explain before coding away from PC
- Easier to see who understands
- Removes syntax/interfaces etc
- Less virtual, more tangible
- Eg use Human robots – Logo
- Use instruction cards / Cut-out symbols

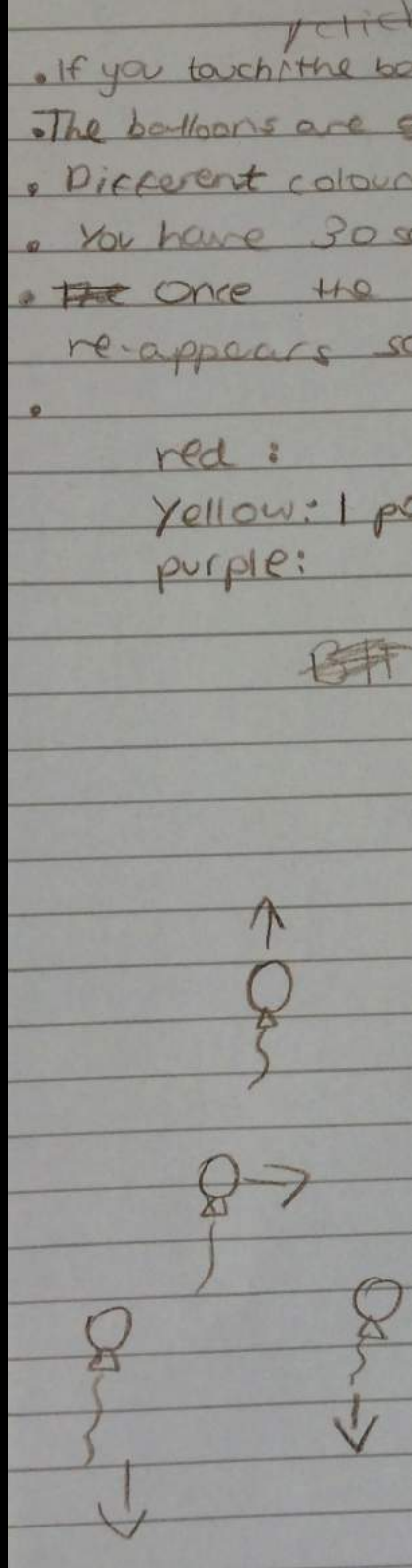


FIGURE 3.1 Children with blind-folds playing turtle

move
move
click
click
click
click
move
Down
move
Hover

Complex Offline Concepts

- Basic – Move Robot to carpet
  - SEQUENTIAL – move fwd 5 ***or***
  - SELECTIVE – move fwd until floor == carpet
- Collect up books:
  - SEQUENTIAL – move fwd 5 pick up move fwd 5 pick up book etc etc ***or***
  - LOOP – [pick up book – add 1 to b – move – if b==30 stop]
- Visual variables
  - Class throw balls of paper into labelled box

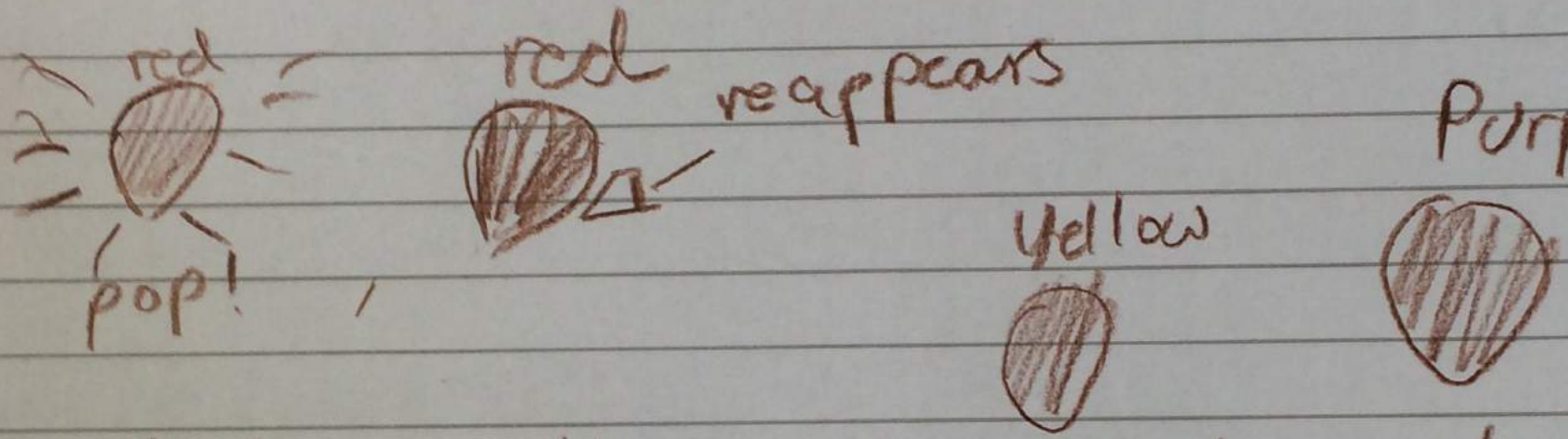
Creating Algorithms by Deconstruction

- Algorithm – a set of steps to solve a problem
- Show pupils a ready made app/program
- Get them to write down the rules they think are making it behave the way it does
- This set of rules is an algorithm
- Now code the app turning your steps into code

[redacted]

- stops after 30 seconds
- at the start the bollones go up
- after you pop the bollones they reapear some were random.
- after you clike the bollon it give you a higher score
- different colours of bollons give different points

- When you click a balloon the score goes up
- After 30 sec, the game ends automatically
- After a balloon is popped it moves in different combinations and starts again in the same position.



The balloon will reapper somwhere else different balloons different position.

Teaching Computational Thinking or Teaching Coding

CT develops skills other than coding:

- Breaking down bigger problems into smaller parts
- An approach to testing things out, experimenting
- Creativity, prediction & perseverance