Computer Science, Maths, Raspberry Pi

Alan Mycroft

Computer Laboratory, University of Cambridge

`http://www.cl.cam.ac.uk/users/am/`

and

Raspberry Pi Foundation

`http://www.raspberrypi.org`

Sheffield Cutlers Ambassadors – 3 Oct 2013

Structure of this talk

UNIVERSITY OF CAMBRIDGE

- Computer Science and ICT
- Raspberry Pi: what and how
- Computing meets Maths

A question I asked school students

UNIVERSITY OF CAMBRIDGE

Question: tell us about the most interesting program you've written.

Answer c. 1990: used a BBC micro to write a program which . . . [Building, Creating]

Answer c. 2010: written a web page and used a package to . . . [Using, Consuming]

During 2000–2010 UK applications for Computer Science degrees halved. Mismatch between student perception and industry demand. School computing (ICT) seemed 'boring'.

[Good news: Cambridge 2013 applications are best ever.]

Royal Society Analysis about School Computing

Three separate concepts:

- Digital Literacy
  Using menu systems, using a web browser, Facebook
- IT (often ICT—Information and Communication Technology)
  Designing big systems, out of components
- Computer Science
  Understanding the basic ideas associated with Computing

The last two are commercially important, and both involve programming. Computer Science focuses more on fundamentals.

Inventing stuff, or consuming the products of Microsoft and Google? (But: re-inventing is good for learning, but not commercially unless you do it better!). Sorting versus using Microsoft Excel?

Did computers get easier or harder to use?

In the last 30 years?

- Easier: they are more powerful, with many great things like Facebook, and amazing graphics.
- Harder: how do I draw a few dots on the screen? Make my computer turns on a light?

Questions: how many computers do you have in your house and car, and how many can you use?

How many in the world?

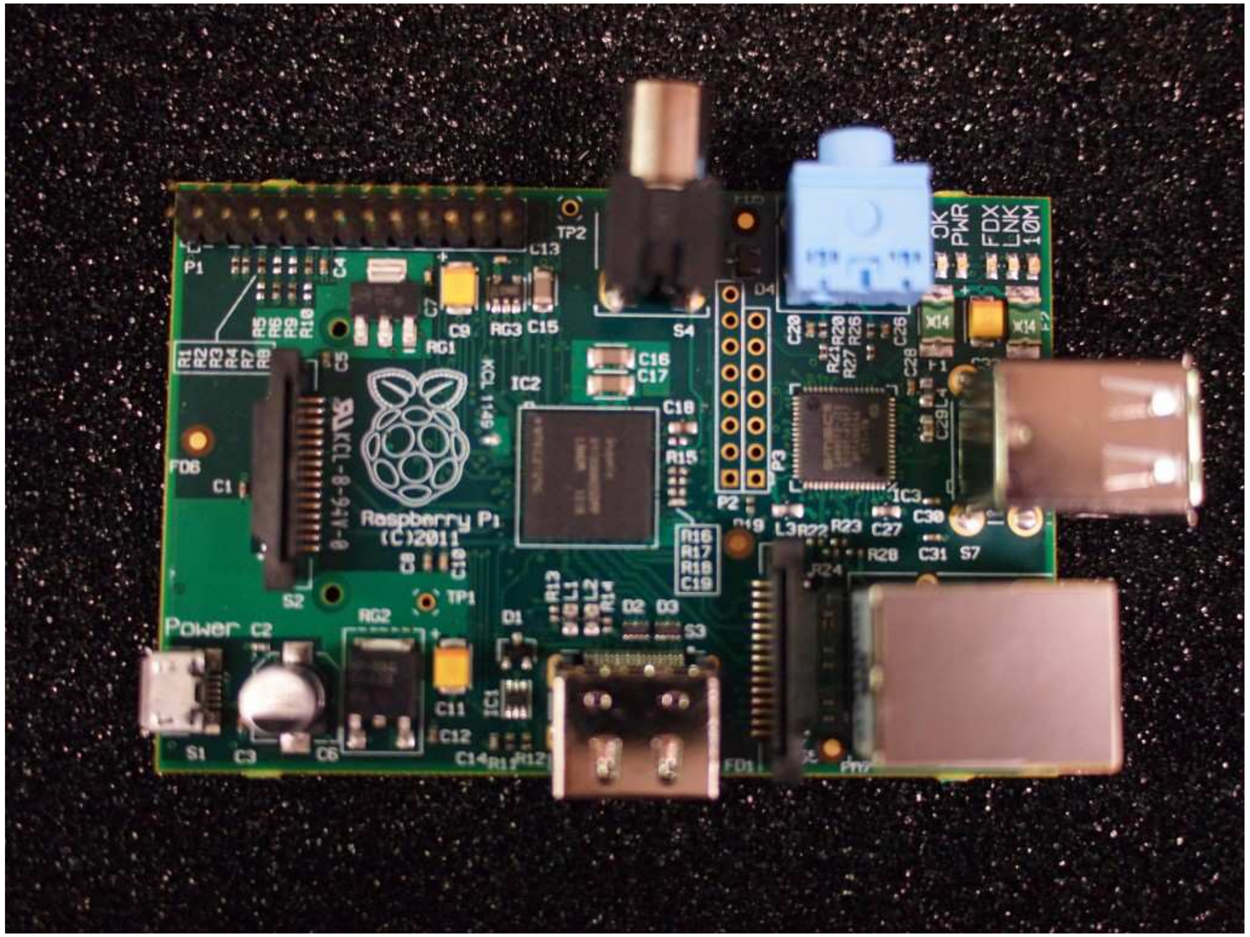
Can you program your phone?

Structure of this talk

UNIVERSITY OF
CAMBRIDGE

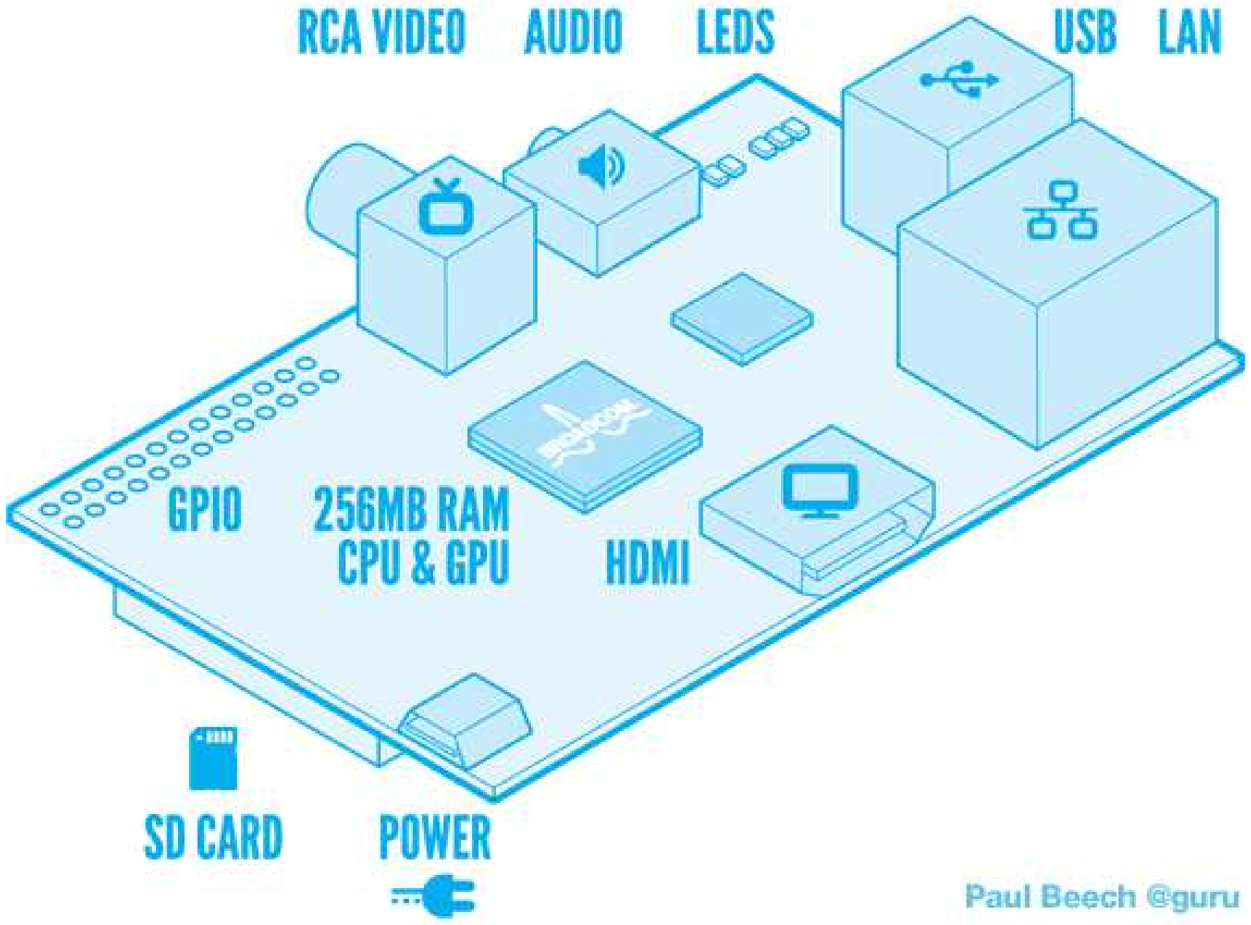
- Computer Science and ICT
- Raspberry Pi: what and how
- Computing meets Maths

Raspberry Pi design (1)

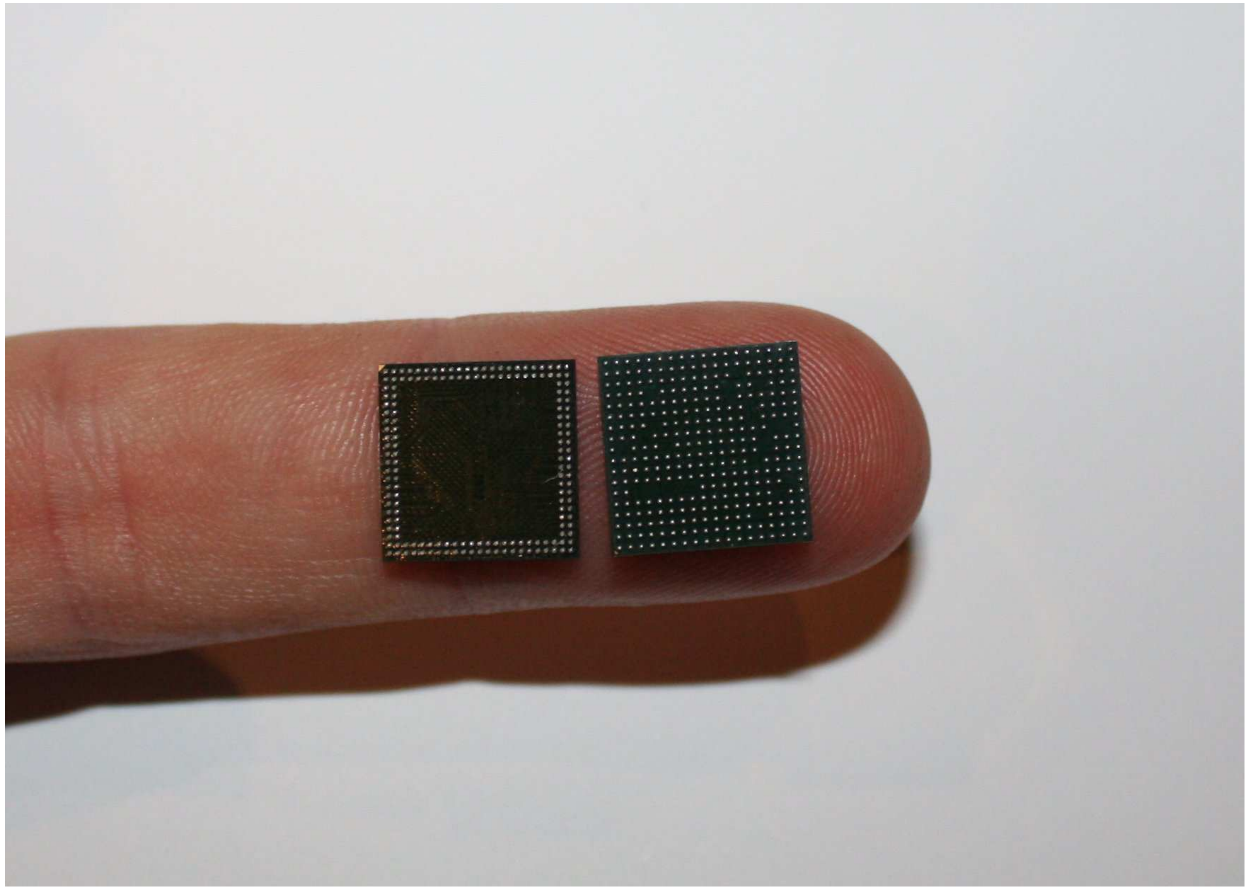


(credit-card size)

Raspberry Pi design (2)



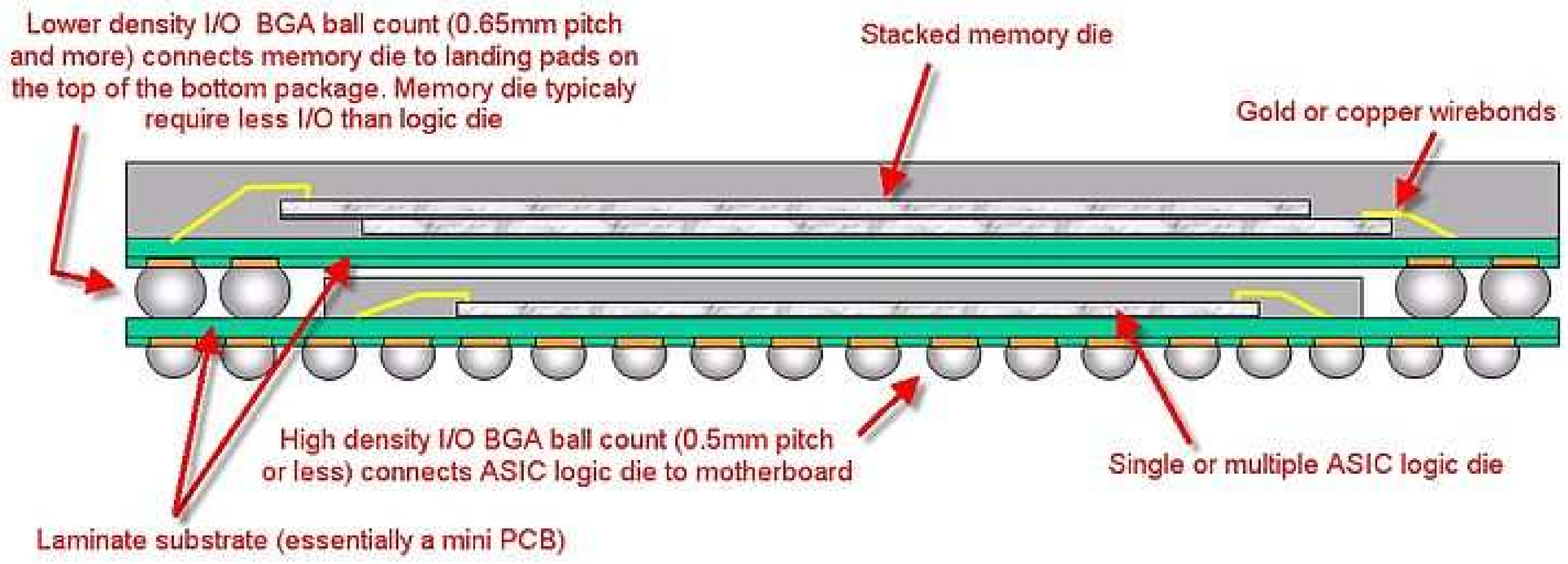
Package on Package (PoP)



Two chips mounted on top of each other.

Memory top (left) and SoC bottom (right) as seen from below.

Package on Package (2)


Lower density I/O BGA ball count (0.65mm pitch and more) connects memory die to landing pads on the top of the bottom package. Memory die typicaly require less I/O than logic die
Stacked memory die
Gold or copper wirebonds
High density I/O BGA ball count (0.5mm pitch or less) connects ASIC logic die to motherboard
Single or multiple ASIC logic die
Laminate substrate (essentially a mini PCB)

[source: Wikipedia]

Etymology

Pi: from the original plan to make Python the operating system, shortened to Pi instead of Py (as everyone knows about π).

Raspberry: computers were traditionally named after fruit: Apple, Acorn [grew into ARM], Apricot, (recent) VIA Banana.

Raspberry has additional subversive/disrespectful/ challenging-of-authority meaning in English (‘blow a raspberry’)!

Why the Raspberry Pi?

UNIVERSITY OF CAMBRIDGE

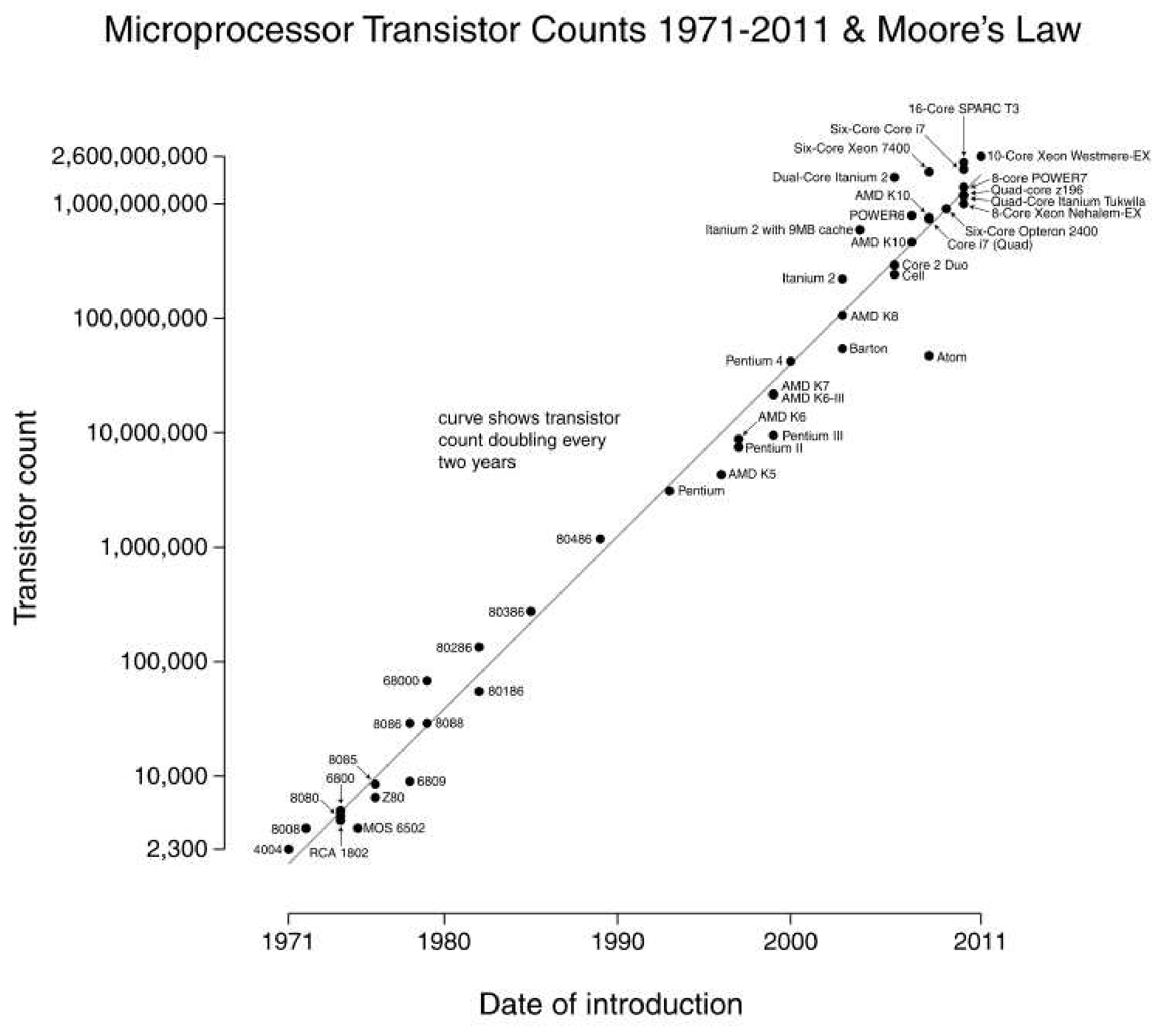
“It’s just an slightly under-powered computer without a screen, and anything you can do on it you could do on a laptop”. Yes, but . . .

- . . . it’s *cheap*, it’s *light* (45g), you can’t wreck it with software
- access to physical pins
- it changes mindsets

The Raspberry Pi Foundation charitable aim is “to promote the study of computer science and related topics, especially at school level, and to put the fun back into learning computing.”

So: if someone starts selling competing kit (e.g. £10, of half the size and twice the memory and processor speed as Raspberry Pi) then this benefits our aims too!

Moore's Law



[source: Wikipedia]

Who would make Raspberry Pi chips?

Remember: fabricating (‘fabbing’) a new chip costs tens of $M.

But: people already had built similar chips for HDTV set-top boxes.

Structure of this talk

- Computer Science and ICT
- Raspberry Pi: what and how
- Computing meets Maths

Let do some programming

How? Do we need a computer? What language?

[For learning programming, we like Python, but I'm going to do "unplugged programming" – to make you think!]

Use an *Algorithm*!

- The essence of a program (a precise description of a process for solving a problem – like a recipe).

Why precise?

- Because computers are stupid (very fast, but very stupid).

How precise?

- like all mathematics – precise enough for the task in hand. We might assume the reader knows (say) long division when explaining some algorithm which uses it.

Practical problem

Take a pack of cards, and sort out the diamond suit into order

$$A, 2, 3, \ldots, 9, 10, J, Q, K.$$

Think carefully: how did you do it?

This is your algorithm!

Our first algorithm

- Problem: given a list of integers as input, produce a list containing the same integers, but in increasing order ('sorting').
- Algorithm: ('selection sort') find the smallest, output it and remove it from the input list. Repeat this process until the list is empty.

An issues

- 'I don't know how to find the smallest'.
  Need to explain down to a level the reader can understand.
  When *programming* the reader is a dumb computer.

UNIVERSITY OF CAMBRIDGE

Finding the minimum of a list

Provided the list is non-empty then:

- Take the first element as provisional minimum.
- For each remaining element (maybe zero of them!)
  - if this element is less than the provisional minimum then use this value as new provisional minimum.
- The provisional minimum now is the result.

This is programming unplugged – without all the typing!

How fast is selection sort?

How many comparison steps?

Suppose we have just 5 cards.

How fast is selection sort?

How many comparison steps?

Suppose we have just 5 cards.

1. Find smallest in 4 steps
2. Find 2nd smallest in 3 steps
3. Find 3nd smallest in 2 steps
4. Find 4nd smallest in 1 steps
5. Only the biggest left: find it in 0 steps!

Total $= 4 + 3 + 2 + 1 = ?$

A bit of maths

UNIVERSITY OF CAMBRIDGE

If we had 101 cards, we would require

$$\text{Total} = 1 + 2 + 3 + \ldots + 99 + 100 = ? \text{ steps.}$$

Who can add this up?

Story about Gauss's teacher…

A bit of maths (2)

$$\text{Total} = 1 + 2 + 3 + \ldots + 99 + 100 \quad = 5050$$

Why: pair off the numbers from each end:

$$1 + 100, 2 + 99, 3 + 98, \ldots, 50 + 51$$

How many numbers?

All are 101.

So total is 5050 (easy?).

Experts only

$$\text{Total} = 1 + 2 + 3 + \ldots + n = \frac{1}{2} \times n \times (n+1)$$

or

$$\text{Total} = 1 + 2 + 3 + \ldots + n = n^2/2 + n/2$$

or

$$\text{Total} = 1 + 2 + 3 + \ldots + n \approx n^2/2$$

So if the input gets 10 times bigger the program runs 100 time slower!

Summary

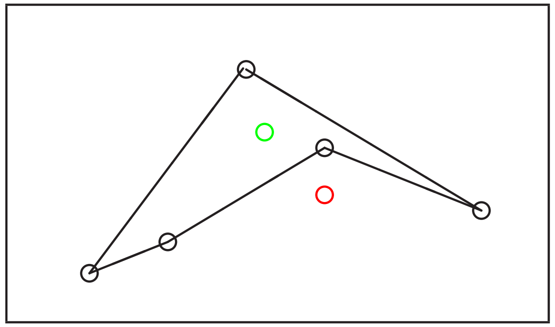
- Playing with computers is fun (building your own games is even more fun than playing others).
- Programming is how you do it
- Algorithms are the core idea behind programming.
- We designed Raspberry Pi to re-attract people to Computer Science.

Taster: The Inside-or-Outside Problem

Given a closed polygon, expressed as a list of points

$$[(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)]$$

then is another point (x_0, y_0) inside it or outside it?



Two-year old humans can do it, but it's really quite non-obvious how to program a computer to do it! It is simple to start by reading $2n + 2$ numbers but then ...???

Wikipedia tells you how, but it needs more maths.