# Computing:

# The Science of Nearly Everything.
# Including Biology!


Aaron Sloman, School of Computer Science, University of Birmingham, UK
http://www.cs.bham.ac.uk/~axs

Presented at
http://www.computingatschool.org.uk/ "TeachShare" on 8th June 2011

Slides here:
http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#cas11

# WARNING and THANKS

## These slides are still rather disorganised and may change.

Comments, criticisms, and especially suggestions for improvement welcome.

A version will be installed on my slideshare.net site.

If requested I can produce a 2-up version for printing two slides per page.

---

Many thanks to

Simon Humphreys (ComputingAtSchool Coordinator, simon.humphreys@computingatschool.org.uk)

and to

Tim Bateson (ICT Strategy Manager, Houghton Kepier Sports College, Tyne and Wear

tim.bateson@houghtonkepier.org.uk)

who together very kindly acted as an audience when I had a trial run using the presentation tools.

Their feedback and suggestions made a huge difference when I eventually delivered a presentation based on some of these slides on 8th June 2011.
a recording of the session is here: `http://www.computingatschool.org.uk/index.php?id=teachshares`
Apologies for sound gap near beginning.

I am also grateful to Simon for all the effort he has put into CAS and for encouraging me to do this presentation when I thought nobody would be interested.

And to Tim for creating the Poplog in linux in virtualbox package which now makes it possible for people (and schools) who are stuck with windows to use Poplog and other linux tools on Windows or Apple Macs as well as on Linux, as explained here:

`http://www.cs.bham.ac.uk/research/projects/poplog/ova/A-README.html`

# What should aims of computing education be?

**Teaching computing should not just involve concerns about:**

- Useful skills of various kinds,
- useful and/or entertaining applications,
- formal properties of computations
- hardware/software engineering,
- serving the needs of the national economy.

**But also:**

- Powerful and deep new concepts and models,
- able to illuminate many other disciplines,
- including studies of mind and life.

That is not obvious to most people, including some of the people with the deepest knowledge of current computer science and engineering.

If even they don't understand the importance of their own field, how can we expect to persuade politicians, educational administrators, exam board, parents and others.

**Start by educating as many children as possible to understand these issues, even if only in their "spare time" activities? They can later help change the world!**

Powerful tools are available to help, and probably a nucleus of teachers willing and able, if only they are given the freedom.

# What should we do?

**We should be doing much more to stretch at least a subset of young minds using these ideas.**

This is not a new idea: it drove the work on

– Logo,

– the One Laptop Per Child (OLPC) project,

– Scratch,

and others.

E.g. Minsky's papers for the OLPC project, summarised here:
`http://www.cs.bham.ac.uk/~axs/teach-share.html#minsky`

and the person who introduce me to AI, just over 40 years ago:
Experiencing Computation: A Tribute to Max Clowes
`http://www.cs.bham.ac.uk/research/projects/cogaff/00-02.html#71`

NOTE:
The assumption that everyone should have the same core education can seriously limit the knowledge we pass on to future generations.

# Two views of the content of computing education

(1) A large collection of skills to be learnt.

This seems to be how government ministers think of education generally!

This is how many people think of a computing education, whether it is IT, ITC or CS.

**Contrast that with:**

(2) A body of knowledge: as in the CAS BOK (Body of Knowledge)

Knowledge about computers, computing, programming, programming formalisms, programs, networks, applications, design, testing, theory, ...

# Are those two educational goals enough?

**No: This is a huge and important field, that needs to face outward as well as inward, providing new ways to think about many things**

**– not just what can be done with computers.**

It is possible to teach computing as a collection of fun things to do and useful practical skills to learn, but this usually leaves out one of the most important reasons to teach computing (including programming) to the **majority** of learners —

I.e. those who are not going to use programming techniques to build something useful or entertaining, but who could use computing ideas and techniques to help them understand the world they live in, including aspects of their own minds.

See the Preface and Chapter 1 of The Computer Revolution in Philosophy (1978)

`http://www.cs.bham.ac.uk/research/projects/cogaff/crp/preface1.html`
`http://www.cs.bham.ac.uk/research/projects/cogaff/crp/chap1.html`

# There is much more to learn about computation than computing skills or knowledge of computing!

Moreover, it is relevant to a (potentially) much larger set of learners.

Examples:

- Learning new ways to ask and think about questions of the form

  "How does that work?"

  – acquiring ever-deepening understanding of the world we live in
  – including ourselves,
  – and awareness of important problems that are not yet solved.

- A computing education can contribute hugely to that understanding,

  because so much of the world involves things, including animals of many kinds, that

  – produce, manipulate, create and use information,
  – of different sorts, processed in very varied ways,
  – which we are just beginning to understand.

- Today's educators need to help produce tomorrow's **thinkers** who will extend our understanding, not just tomorrow's **application developers**.

- Moreover, we can't predict detailed future requirements, so the enriched computing education must be diverse, not standardised.

The "design-based" approach to the study of complex information processing systems can be contrasted with the work of scientists who merely observe, measure, and produce correlations and graphs.

For more on the "design-based" approach to the study of mind see
`http://www.cs.bham.ac.uk/research/projects/cogaff/misc/design-based-approach.html`

# Example: What is the universe made of?

It is clear that a great deal of what we find in the universe is composed of, and involves transformations of,

- matter
- energy, and
- information.

It's not just man-made machines that process information.

Not all information processing systems use bit patterns implemented in transistors.

All biological organisms use and deploy energy in what they do.
The processes involve selection from options and control of details.

Informed control: both selections among alternatives and online modifications of behaviours are all based on **information**, about: needs, opportunities, constraints, achievements, discrepancies, resources available, structures, processes, opponents, helpers ....

The information can come from various sources, e.g. from external and internal sensors, from things learnt previously in the individual's life, by reasoning, and from the genome.

**We need to educate far more people so that they can think about the workings of information-processing systems, on the basis of first-hand practical experience of designing, building, testing, debugging and explaining (initially simple) examples:**

This should be a pre-requisite for studying and teaching biology, psychology, neuroscience, philosophy ... and several other subjects. It can also help mathematicians explore mathematical spaces.

# Living things are informed control systems

Multiple designs are found in living things.

Not only varied designs for physical structures, from microbes to elephants –
but also a wide variety of ways of processing information at the level of

- cells (e.g. detecting intruders or repairing damage)
- whole organisms (e.g. selecting actions, controlling movements)
- groups of organisms – e.g. foraging insects

Living things are information processors: informed control systems

**Organisms are control systems:**

They acquire and use information (external and internal),
in order to control how they use matter and energy

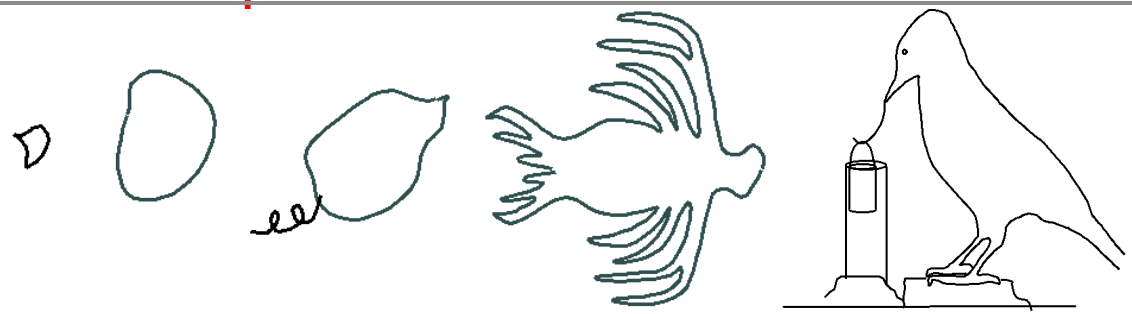
(in order to acquire more matter, energy and information,
and also reproduce, repair, defend against intruders, dispose of waste products...).

Over many millions of years, evolution produced

- more and more sophisticated information processors,
- driven partly by changes in physical environments,
- partly by results of prior evolutionary history
- in their ancestors and in other organisms (enemies, food, symbiants, ...).

Much of that process is unexplained: people studying evolution have not learnt to think about possible requirements for information processing systems, and possible designs.

# All organisms are information-processors but the information to be processed varies and so do the means



## Types of environment with different information-processing requirements

- Chemical soup
- Soup with detectable gradients
- Soup plus some stable structures (places with good stuff, bad stuff, obstacles, supports, shelters)
- Things that have to be manipulated to be eaten (e.g. disassembled)
- Controllable manipulators
- Things that try to eat you
- Food that tries to escape
- Mates with preferences
- Competitors for food and mates
- Collaborators that need, or can supply, information.
- and so on .....

Discussion topic: How do the information-processing requirements change across these cases?
Discussion topic: How does consciousness fit into these ideas? Self consciousness?
Discussion topic: Look up “Betty Crow Hook” and discuss (Oxford research reported world-wide in 2002)

# Using Computation to Illuminate

Instead of always teaching programming because of its practical applications, we can also teach some learners to use computing ideas and programming experiments to illuminate other disciplines.

Especially helping us think about natural intelligence, in humans and other species

[Including re-shaping questions about the nature of mind and consciousness.]

Example topics for learners – What forms of computation are involved in

| | | |
|---|---|---|
| Using language | Visual perception | Analogical reasoning |
| Motivation | Emotions | Planning |
| Plan execution | Learning | Evolution of consciousness |
| Curiosity | Play | Developing numerical competence |

and many more... ?

For example:

- Many learners like creating programs that manipulate fragments of natural language, initially in textual form, and initially in very simple ways (as in "toy" chatbots), then gradually adding deeper models of uses of language to talk about something.
- We can show how Darwin's thinking (and thinking of some of his opponents) about evolution of consciousness might have been different if he had known about virtual machines in computers:
  `http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#darwin`

What else? Biology, psychology, economics, linguistics, philosophy, mathematics.

The remaining slides illustrate only a small subset of the possibilities, largely from an AI standpoint.

# A new view of the general principles

Denning and his colleagues, among others, have attempted to collect ideas about what the "fundamental principles" of computation are, e.g. to feed into educational systems, or curricula.

E.g. `http://cs.gmu.edu/cne/pjd/GP/gp_overview.html`

But those general principles are distilled from a relatively narrow body of knowledge and experience, focused on relatively old ideas about what computers can be used for

(and, I would argue, focusing only on a restrictive subset of those things – e.g. a limited sample of types of programming language and types of information processing system).

We need to include, among the fundamentals, things like

- What kinds of functions information processing systems (IPSs) can have (including all the biological control and communication functions).
- What sorts of information contents IPSs may need to operate on or use – e.g. what kinds of things, properties, relations, processes, causal interactions, agents, intentions, plans, etc. such IPSs may need to observe, create, hypothesise, monitor, act on, etc.: what ontologies are used.
- What forms of representation they can use for encoding those different kinds of information.
  e.g. sets of numbers, matrices, equations, parse trees, graphs, neural nets, grammars, diagrams, ....
- What mechanisms (including algorithms) can be used to operate on and with those representations.
- What sorts of sub-systems can be combined, and how they can be integrated, in various architectures
- What kinds of environment such IPSs can interact with, using what sorts of sensors, effectors or communication media, and what they can try to achieve, preserve, avoid or prevent in that environment.
- How an IPS can grow or change its architecture, mechanisms, forms of representation, ontologies, functions, knowledge and interactions, and how a genome can specify how changes can occur.
- How alternative designs and implementations can be compared and evaluated, in relation to their functions, in various environments. **This is one of the most important things to teach.**

# What Fred Brooks got wrong

Brooks has made major contributions to computing, and is highly respected and often quoted. However one of the often quoted portions of his 1994 ACM Allen Newell Award lecture "The computer scientist as toolsmith" is mistaken. He wrote:

> "Perhaps the most pertinent distinction is that between scientific and engineering disciplines. That distinction lies not so much in the activities of the practitioners as in their purposes. A high-energy physicist may easily spend most of his time building his apparatus; a spacecraft engineer may easily spend most of his time studying the behavior of materials in vacuum. Nevertheless, the scientist builds in order to study; the engineer studies in order to build. What is our Discipline?
>
> "I submit that by any reasonable criterion the discipline we call "computer science" is in fact not a science but a synthetic, an engineering, discipline. We are concerned with making things, be they computers, algorithms, or software systems."
>
> `http://www.cs.unc.edu/~brooks/Toolsmith-CACM.pdf`

That view of the field fails to take account of all the points I have been presenting, and ignores a great deal of the scientific work being done in computer science departments in many countries, including the scientific work that was the main focus of Allen Newell, in honour of whom the award was named.

Newell, and others, have tried to understand cognition as computation.

More generally, there is now much scientific investigation of biological information processing and that work requires the expertise of computer scientists, since the other disciplines involved lack some of the important concepts and theories.

However, it is likely that there will be new cross-disciplinary scientific sub-fields, as happened in biochemistry and biophysics.

These are important points for the future work of the Computing At School group (CAS). for whom this presentation was prepared: `http://www.computingatschool.org.uk/`

# Topics for possible discussion within CAS

1. Do we want to broaden the scope of CAS to include:

   teaching about the ways in which computing ideas and programming experience can illuminate other disciplines,

   especially understanding natural intelligence, in humans and other species?

   [Including the nature of mind and consciousness.]

2. How do those goals affect the choice of computing/programming concepts, techniques and principles that are relevant?

3. What are good ways to do that?

   E.g. what sorts of languages and tools help and what sorts of learning/teaching activities? How much freedom should teachers have to choose what suits them and their pupils?

4. Which children should learn about this?

   Contrast

   - Offering specialised versions for learners interested in biology, psychology, economics, linguistics, philosophy, mathematics, etc...
   - Offering a study of computation as part of a general science syllabus.
     (Better to start with this then move to specialised versions.)

5. Is there any scope for that within current syllabus structures, and if not, what can be done about making space?

# Beginners (and their teachers) need powerful systems

There are implications regarding programming languages and tools.

In particular, for some forms of education it is not enough to think of the programming needs of the pupil: gifted teachers will have great ideas about things that require development of languages, tools or packages that provide new launchpads for student learning, as described in a paper published in 1984:

Beginners need powerful systems (A tribute to Max Clowes)
`http://www.cs.bham.ac.uk/research/projects/cogaff/81-95.html#45`

Example requirements:

- Support for developing new micro-languages suited to specific domains:
  E.g. the language for programming grammars, or the river world
  `http://www.cs.bham.ac.uk/research/projects/poplog/teach/river`
- Support for making the new code libraries and documentation available to users in a way that integrates with the rest of the system being used.
- That can include supporting re-use, e.g. supporting the development, by teachers, of new packages that are themselves part of the background implementation layer for further new packages.
  (E.g. making a natural language package or database package or reasoning package available for integration in new teaching materials.)

These were all motivations that drove the development of Pop-11 and Poplog at Sussex university, at the same time as it was being developed as a commercial product for industrial use.

Much of this thinking was inspired by Max Clowes, who unfortunately died in 1981 after helping to set up teaching and research in AI at Sussex University, from about 1969.

# Remaining Slides

- The remaining slides present examples of topics that illustrate the preceding points
- about teaching young learners how to think about natural as well as artificial information processing systems
- using ideas derived from attempts to build working models as well as problems encountered in attempts to build human-like or animal-like machines.
- The list is merely illustrative: different teachers will prefer to focus on different topics.
  (There is great scope for teacher creativity.)
- Some of the examples are fairly sophisticated and would only be suitable for advanced learners.
- Others are very simple and could even be used in primary school.
- This is a personal list, based on my experience of teaching AI in various contexts: a lot more work is required to produce a broad list of examples to help teachers choose, or design their own topics.
- I have used examples based on a programming language (Pop-11) and development environment (Poplog) that I know provides powerful tools for teachers and learners. There is no implication that this combination is the only option: it merely illustrates what is possible.

# Programming “bumpy” agents vs “thinky” agents

There are now powerful tools enabling learners to program simulated agents moving around in a simulated world where they sense other things in the world and react accordingly, e.g. by producing simulated physical changes in their behaviour, such as

- shooting
- chasing
- bumping into
- pushing
- following

A quite different sort of task is to develop simulated agents that acquire store, use, share, and later modify information obtained from the environment.

An example is a simulated agent that explores some terrain and builds up a map of where things are which it can later use to make plans of various sorts involving taking, fetching, visiting, etc.

This could be extended to include cooperative or competitive behaviours, e.g. sharing information about where food or dangers or shelter are to be found, possibly entering into trading arrangements.

This will require some thinking not only about what information is acquired and used, but also what sorts of motives the agents can have, and how their motives are generated, how conflicts are detected and resolved (e.g. between low short term gain and high long term gain).

Exercises of this sort could lead to discussions about how societies evolve, the function of money, the need for law-makers and law enforcers, etc.

# Use of language involves information processing

Learners can enjoy getting computers to manipulate fragments of their own language.

- Initially a child can play with several "chatbot" programs, such as this (rather simple) eliza : `http://www.cs.bham.ac.uk/research/projects/cogaff/eliza`

- And then later learn how to build such a thing
  - starting with a very simple version,
  - Then later adding more sophistication,
    e.g. use of a simple grammar.

- For this, it is a great help to have a programming language that
  - includes facilities for list processing (including **lists** of strings, instead of only strings)

  and which includes
  - a pattern matcher for recognising, decomposing and re-creating patterns in list structures.

    One example is the Pop-11 pattern matcher:
    `http://www.cs.bham.ac.uk/research/projects/poplog/teach/matches`

The enjoyment and learning that come from getting a machine to manipulate language does not depend on producing a perfect conversationalist.

Analysing and reducing the imperfections of a flawed program can provide deeper learning than simply producing a bug-free program.

A good teacher will set traps that produce buggy programs:
experience of fixing one's own buggy thoughts and designs is one of the most important forms of learning.

# Beyond “surface” manipulation

Here’s a sample interaction with the aforementioned Eliza

(Human types in lower case, and machine responds in upper case.):

? I would like to teach you to swim.
WOULD YOU REALLY LIKE TO TEACH ME TO SWIM
? Yes you would enjoy it a lot
WOULD THIS ENVIRONMENT MAKE YOU HAPPY?
? We can’t swim here
DID SOMEONE TELL YOU YOU CAN’T SWIM HERE
? There isn’t any water here
COULDN’T THERE REALLY ANY WATER HERE ?
? Can you see any water here?
I WISH I COULD SEE ANY WATER HERE
? if there were water here we could learn to swim
HOW CAN IF THERE WERE WATER HERE YOU LEARN TO SWIM ?
? Your grammar is pathetic
PATHETIC ? – NO NOT MY GRAMMAR .
? Yes – your grammar
DIDN’T YOU PREVIOUSLY SAY YOU WOULD LIKE TO TEACH ME TO SWIM

Could a child playing with such a system come up with theories about how it works, and perhaps test those theories, without looking at the code?

# Beyond “surface” manipulation

Eliza-like systems merely look at and manipulate patterns of words, whereas natural language processing techniques go beyond that.

including using generative grammars to parse or produce novel sentences.

It is possible to introduce learners to ways of getting a computer to use a grammar and a lexicon

- to analyse simple sentences in accordance with the grammar
- to generate sentences (e.g. randomly) using a grammar and a lexicon
- and later on to relate grammatical structure to semantic content so as to be able to answer questions about some simple world, or obey commands to act in the world

  As Terry Winograd demonstrated around 1971 at MIT:
  A simplified version of his model (SHRDLU) is shown in this online video (GBLOCKS):
  `http://www.cs.bham.ac.uk/research/projects/poplog/figs/simagent/#gblocks`

  Students can be given the package and invited to modify the grammar, and/or the lexicon, to allow more varied forms of interaction, or to enrich the initial environment. Some snapshots are below.

  Less experienced learners could be given a partially built (possibly buggy) simpler version of this program and invited to extend it (or debug it – looking only at a small subset of the code).

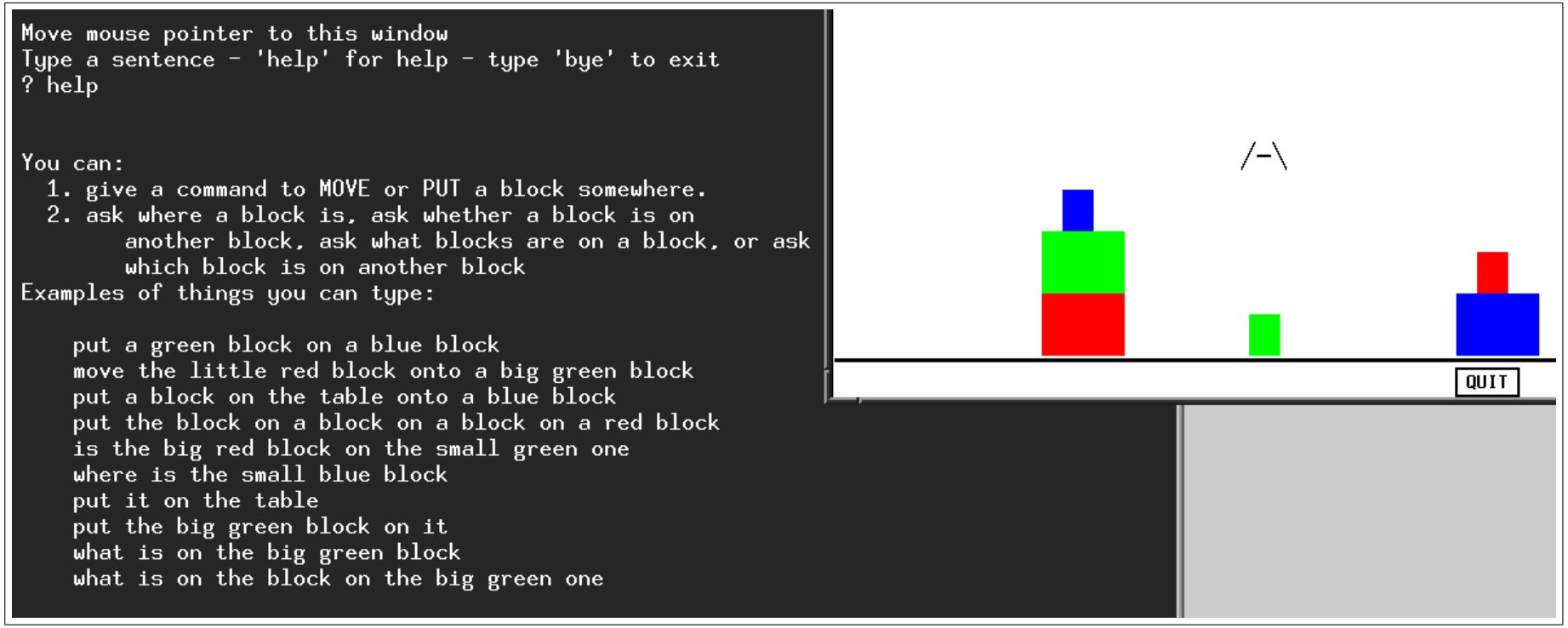
  An example of that approach is the move from this exercise
  `http://www.cs.bham.ac.uk/research/projects/poplog/examples/#river`
  to this one:
  `http://www.cs.bham.ac.uk/research/projects/poplog/teach/riverchat`
  (among many other possibilities).

# Snapshots of a toy version of SHRDLU running

This video shows a “conversational” simulated robot inspired by Winograd’s SHRDLU:

`http://www.cs.bham.ac.uk/research/projects/poplog/figs/simagent/#gblocks`

Here is a snapshot from a run of the program:



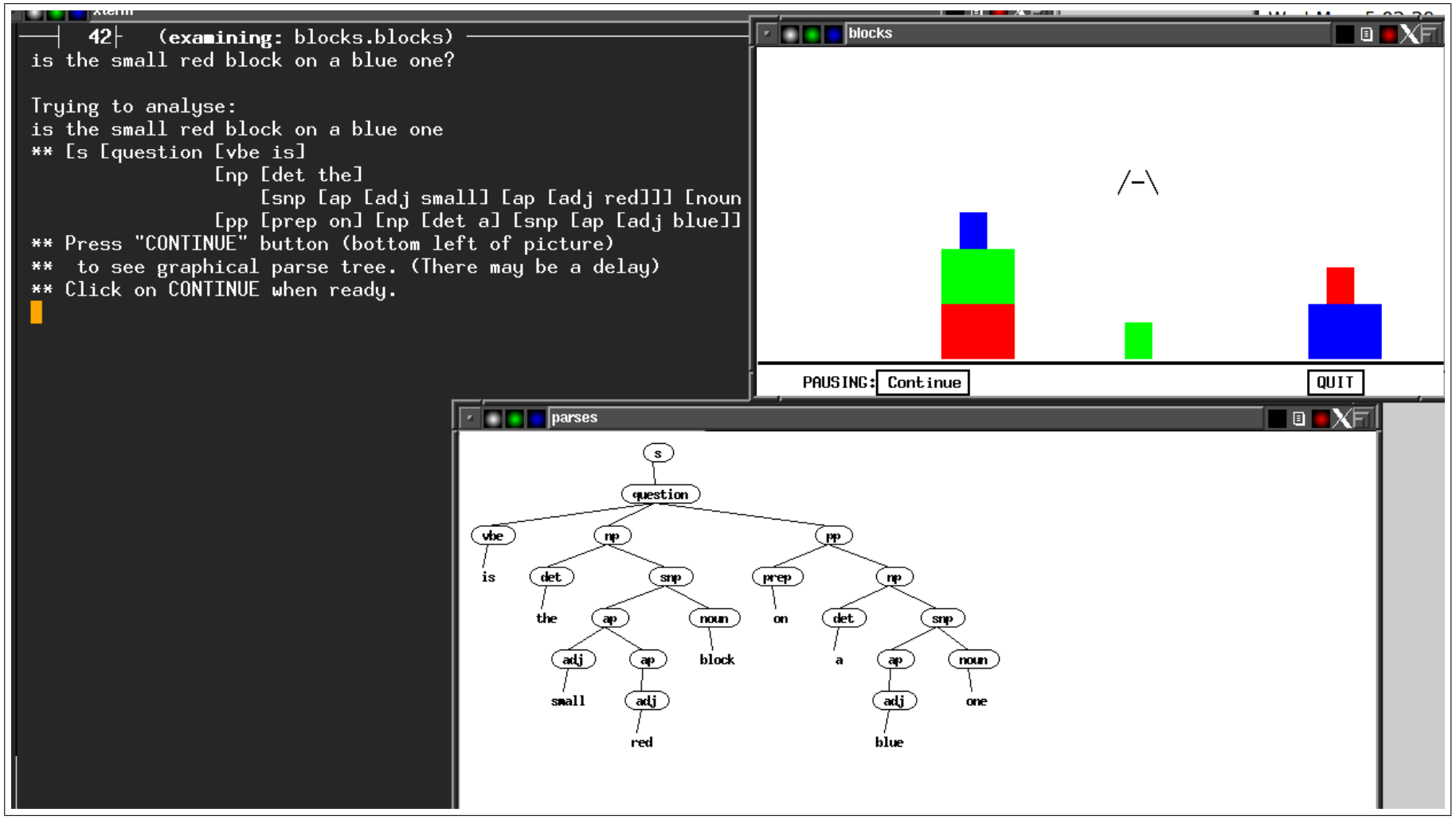

The picture shows a simulated robot hand above a table on which there are six blocks, of two sizes and three colours.

The ‘help’ command gives instructions, as shown.

Several more examples follow.

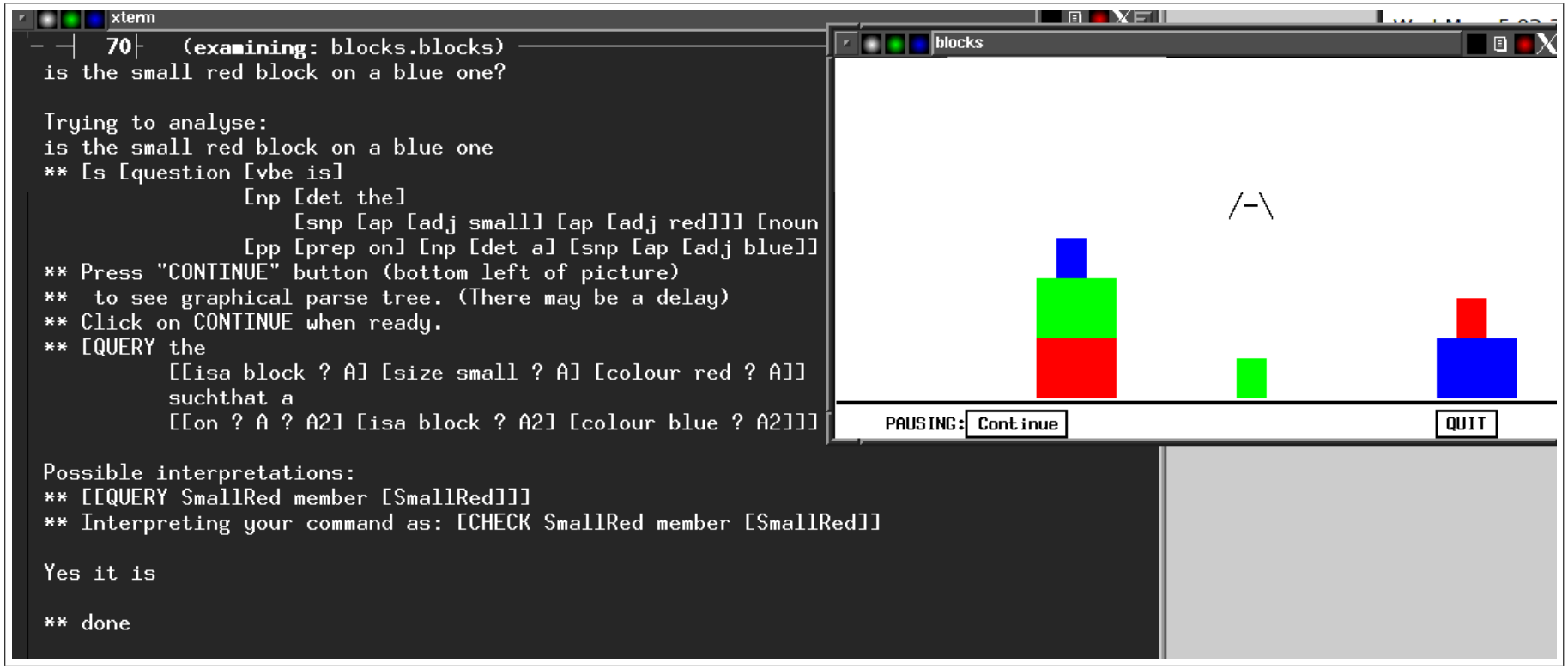
# The Toy SHRDLU - snapshot 2

The parse tree for “is the small red block on a blue one” is shown both as a textual representation of a list structure (list of lists of lists...), on the left, and as a picture, below.



# The Toy SHRDLU - snapshot 3

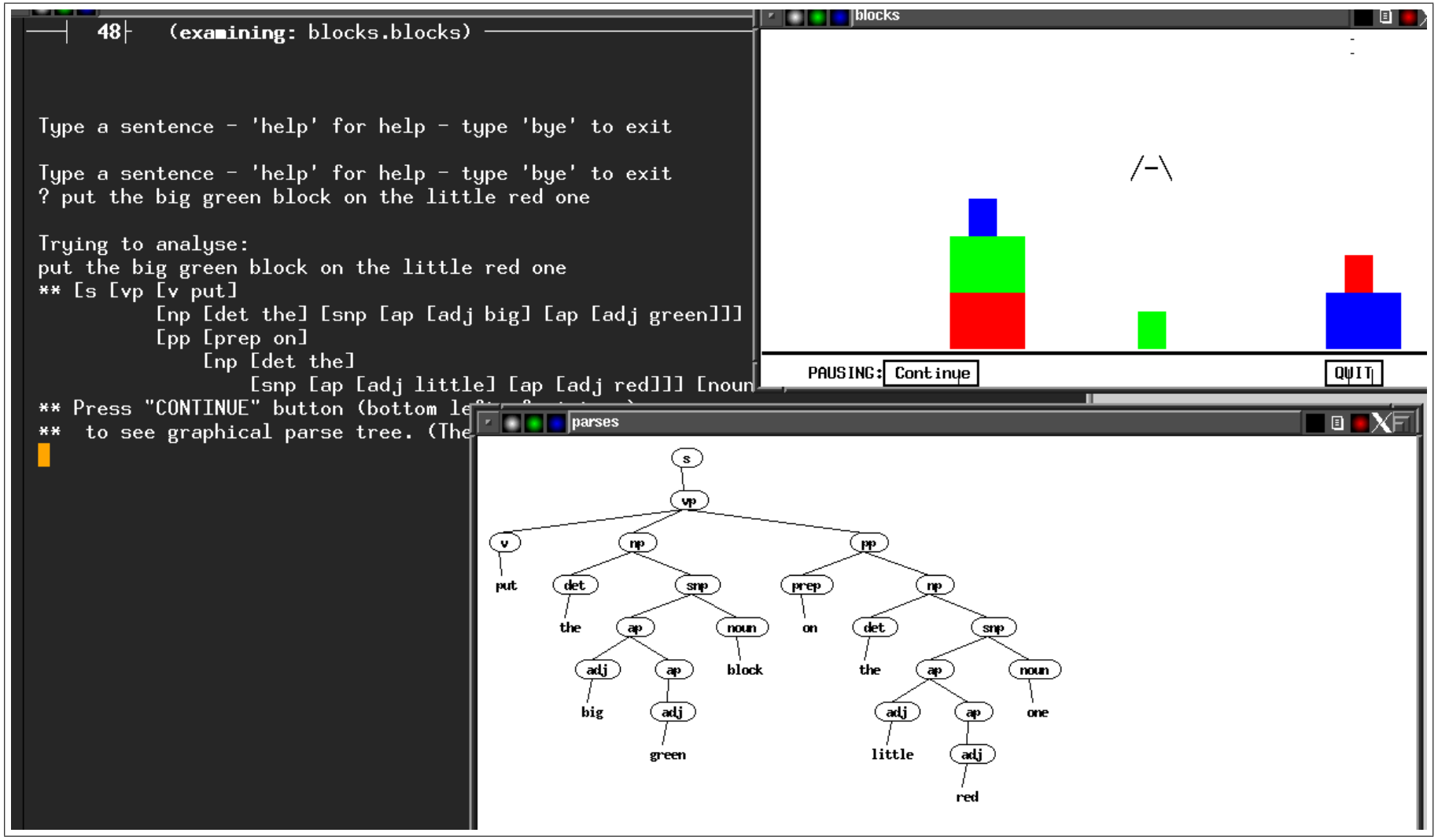
It goes from the parse tree to a “semantic interpretation”, working out which entities in the known world (represented as a simple database of facts) might be referred to in the sentence, then answers the question if it is unambiguous:




It interprets the question “is the small red block on a blue one” as being about what is in the set of blocks that are on blue blocks, and decides that the block known as SmallRed is a member of the set of blocks that are on blue blocks.

# The Toy SHRDLU - snapshot 4

Now showing the result of parsing “Put the big green block on the little red one.”




From the parse tree it derives an interpretation of the sentence as a command to achieve a new state of the world: the goal state, then finds a plan to achieve the state, by moving blocks around (see the video).

# The result of carrying out the plan to achieve the goal.


(Actions of the program are shown in the SimAgent video mentioned above.)

# More examples of language processing

If learners can be provided with a library package that can accept a user-provided grammar and lexicon and on that basis construct

- a sentence generator
- a sentence parser

then they can do “programming” by specifying a grammar and a lexicon.

Example exercises could include:

Create a grammar and a lexicon for

- railway station announcements
- insults
- the opening sentence of a child’s story book
- simple stories
- haikus

Typically people who try this find that some of the sentences generated by the program were not supposed to be generated, e.g.

Platform five will depart from the London train in five minutes.

Trying to work out how to avoid the unwanted sentences involves thinking about adding more details to the grammar, e.g. more syntactic categories.

Using the student’s grammar to parse the unwanted output can be revealing.

**All that can help students understand better how their own language works.**

# Playing with, modifying, assembling program fragments

It would be unreasonable to expect beginners to be able, even if working in teams, to build up anything like the SHRDLU program starting from just a programming language,

even a powerful AI language, like Lisp, Prolog, or Pop-11.

- An alternative is to demonstrate the whole system, but to give them carefully designed tasks involving parts of the system.
- So a library for doing things with simple grammars can be provided, and students can play with it and try to develop an interesting grammar for communication with a robot – testing the grammar in terms of what it can generate randomly and how it parses test sentences.
- A few, who progress rapidly, may ask to see the parsing and generating code and try to extend the libraries, or produce their own versions, while others stick with the simpler tasks.
- Likewise a planning program could be provided for students to play with.
- They may learn by testing it with various problems and see where it fails.
- A few may try to extend, or modify, the program, produce a new version from scratch.

The opportunities for creative teachers are vast – if they have powerful tools and exemplars to start with. (More are becoming available.)

# Visible and invisible behaviours

Many toolkits are designed to enable beginners to produce simulated agents with visible behaviours, which may include moving around on a 2-D surface, rotating in 3-D, changing colour, changing direction or speed of motion, changing size or shape, etc.

In contrast, many organisms do information-processing that is not externally visible – though some of the effects are.

In the case of humans that can involve processes like

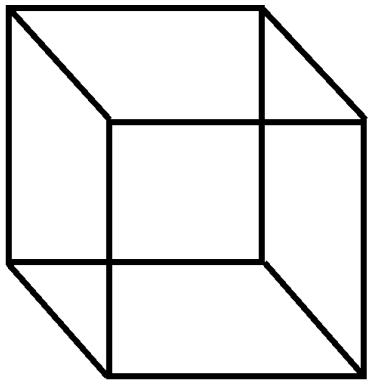
- passive perception
- thinking about something
- making plans
- solving problems
- choosing between alternatives
- learning (e.g. analysing what was done wrong in a recent action).

I'll give some examples of apparently passive perception, where there is no external behaviour but complex internal processes occur.

# Examples of "invisible" information processing

Being **conscious of something** involves processing information

**A familiar example: The Necker Cube**



Although you are looking at a 2-D configuration of black pixels on a white background, you see lines not pixels and you probably see a 3-D structure not just a 2-D pattern. How???

If you stare at it long enough the experience you have will change:

The 2-D lines remain where they were, but you will see them as edges of a 3-D wire-frame cube.

The image is ambiguous and your percept of it can flip between two different views, with edges changing their relative distance from you, and their 3-D orientation, even though nothing changes in the image.

What forms of processing can give a machine the ability to look at a 2-D pattern of marks and interpret the marks as representing a 3-D structure?

What processes in the machine could make the interpretation flip while the input remains unchanged?

If you look at this web site

`http://www.math.ubc.ca/~morey/java/rotator/rot.html`

You will see several more things that can be interpreted as moving 3-D structures, even though only a flat screen is in front of you.

How can static and moving 2-D patterns be interpreted as 3-D structures and processes in the environment?

Some things are easier to model than visual experiences....

# A challenge for the class-room

Organise a discussion on what would need to go on inside a machine to enable it to take in a 2-D visual structure like the picture of the necker cube and interpret it as a 3-D wire-frame object that could not be accommodated on a flat sheet of paper, as the picture can.

- What would have to change inside the machine when the picture “flips” ?
- Are there other kinds of ambiguous pictures? Would the same mechanisms work for all of them?
- Teacher: construct some simple programming exercises involving taking in pictures made of lines and describing them, including pictures with grouping ambiguities, or figure-ground ambiguities.
- A very simple case: a rectangular array of dots could be seen as a collection of rows of dots, or a collection of columns of dots.

  Compare verbal ambiguities: She hit the man with a fish?
  Which man did she hit? What did she hit the man with?
- How should such a program describe an ambiguous picture?
- Some simple examples of getting programs to see structures in pictures made of dots are here:
  `http://www.cs.bham.ac.uk/research/projects/poplog/packages/current/teaching/teach/seepics`
  (I need to produce examples of those programs running.)

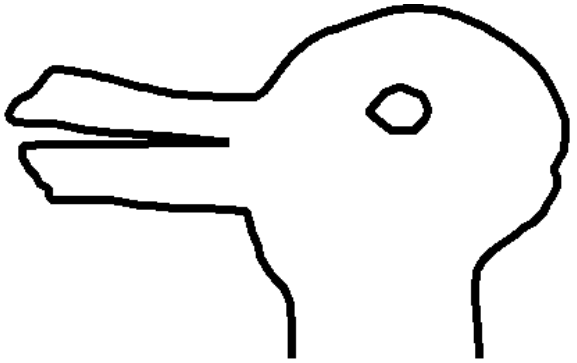
# More examples of "Seeing as"

**The Duck-Rabbit** (Compare the Necker cube mentioned previously.)

There's a 2-D pattern as before, but you probably see this one as an animal with particular physiological features.

If you stare long enough it should flip between two animals with different features, even though there is no change in either the 2-D or the 3-D geometry of what you see.

E.g. a part of the image can be seen as ears, or a bill:
but both alternatives occupy the same 3-D space.



An even more abstract change probably occurs: you see the animal facing to your left or facing to your right depending whether you see it as a duck or as a rabbit.

How can a machine be made to see something as facing left, or as facing right?

The machine would have to have an ontology that includes things having states that involve facing, moving, seeing, etc.

It might relate the direction in which something is facing to what it can see, and which way it is likely to move.

This requires having the ability to think about what **could** happen even if it is not **actually** happening now.

**How would such possibilities, and constraints on possibilities be represented in the machine (or in an animal's mind).**

How can we find out how animals (including humans) represent such information?
Will trying to build a working model help? (Often? Rarely? Never?)

# More visual information processing – for planning actions

Human vision includes many capabilities that are proving very difficult to replicate in current machines.

This may be because of our limited understanding of what the problems are.

(a) (b) (c)

You can look at the scene in (a) and think about how your fingers might have to move to produce the arrangement (b) (using your right hand, or your left hand, or both).

You can also think about a sequence of finger movements that would be able to perform the reverse rearrangement, from (b) to (a).

Current robots cannot do that: What would have to go on inside them to make it possible?

It's not too hard to get a machine to derive image (c) from image (a): could that be a first step???

Notice that examples like this show that seeing is FAR more than attaching labels to portions of images – which is all that some current "vision" systems can do.

Seeing should not be confused with recognising: you can see things you don't recognise.

# A simple program to explore geometric process patterns

This example is a program that can be used by learners to explore ways of generating a variety of geometrical structures by varying parameters in a very simple drawing program (originally inspired by the Logo Turtle).

The key idea is this program, expressed in Pop-11, but equally simply expressible in several other languages:

```
define polyspi(side, inc, ang, num);
    ;;; Draw a polygonal spiral. Initial arm length is side.
    ;;; inc (positive or negative) is added to side at each turn.
    ;;; The angle turned (to left) is ang (in degrees).
    ;;; The total number of sides is num.
    repeat num times
        draw(side);
        turn(ang);
        side + inc ->side;
    endrepeat;
enddefine;
```

It should be obvious that this command will draw a square of side 400:

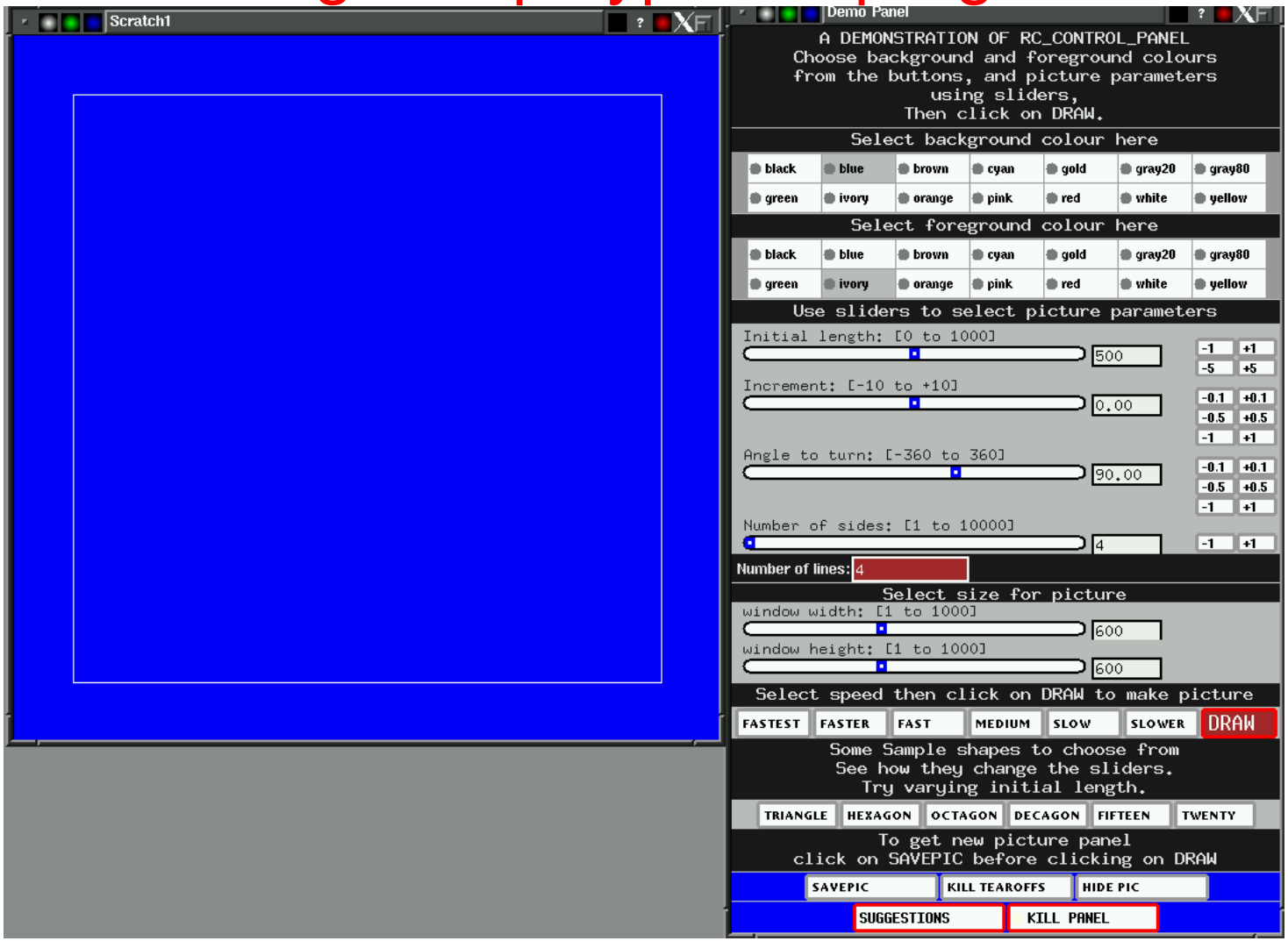
```
polyspi(400, 0, 90, 4);
```

What will these do?

```
polyspi(400, 0, 60, 4);
polyspi(400, 0, 120, 4);
polyspi(0, 5, 60, 100);
```

A program is available to encourage exploration of the space of possibilities.

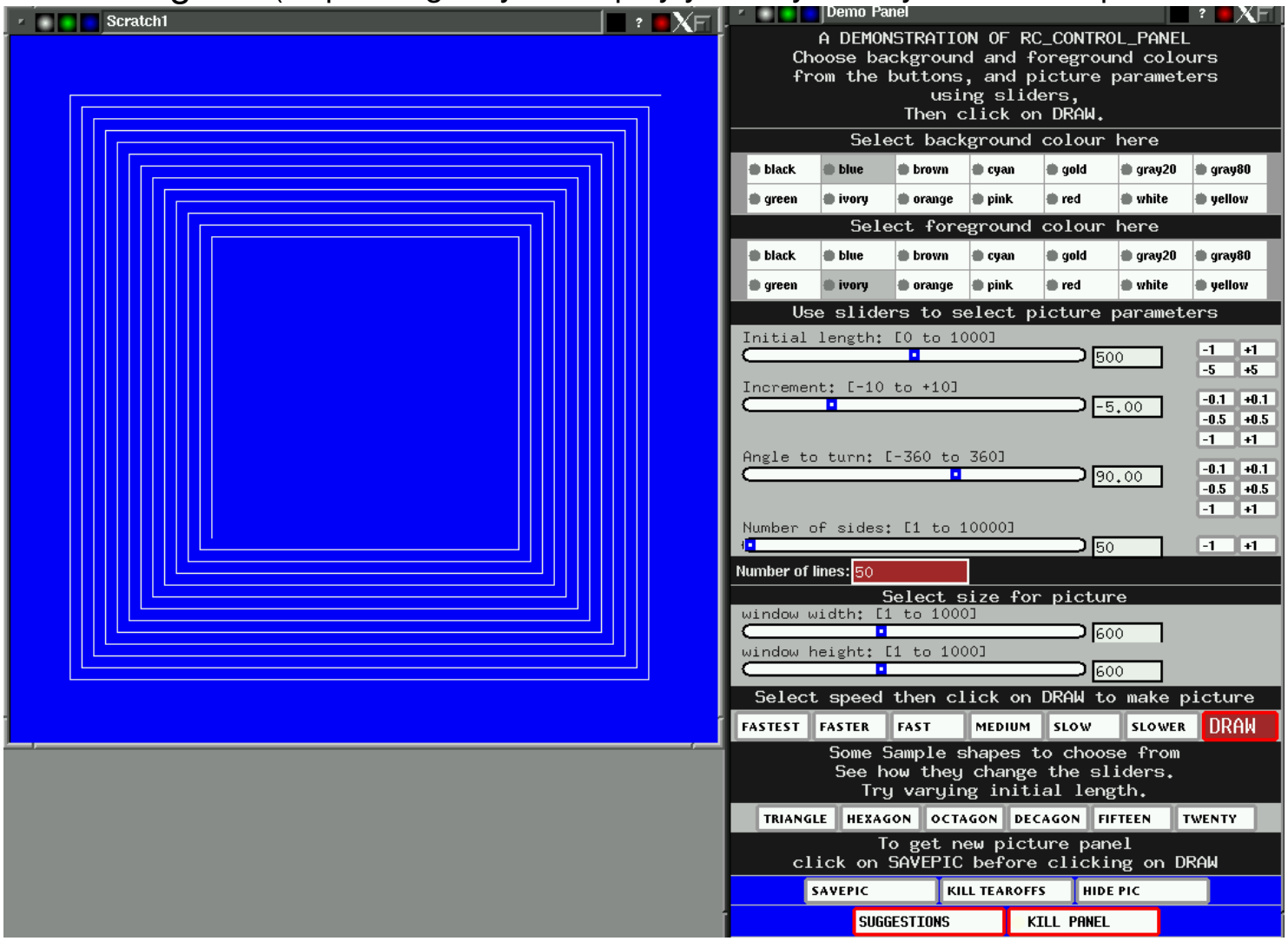
# Using the “polypanel” program




The user can choose background and foreground colours, an initial length, an increment (positive or negative), an angle to turn, a total number of sides, a drawing speed, or a pre-configured example shape, or ask for suggestions. Choices here: blue, ivory, 500, 0, 90, 4

# Classroom use of the polypanel program 1

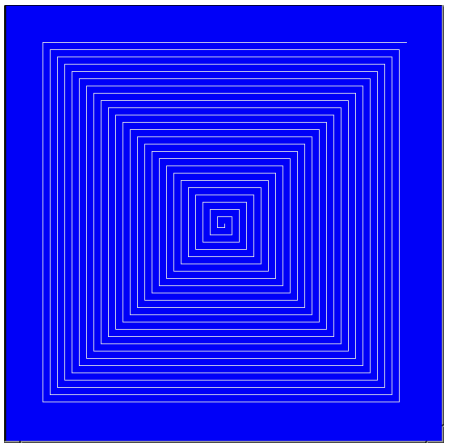
Can you tell by looking at the figure on the left, how the numbers selected in the panel menus were changed? (Depending on your display you may or may not see the picture as intended.)



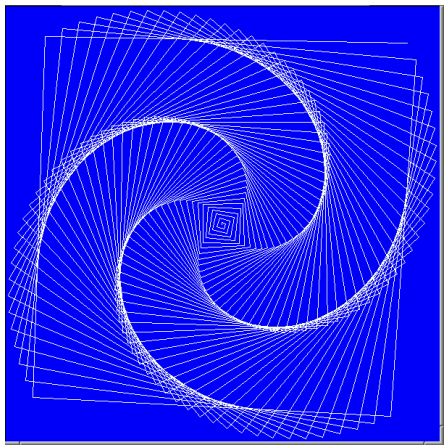
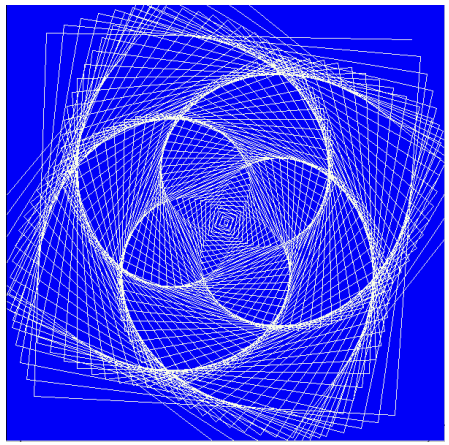
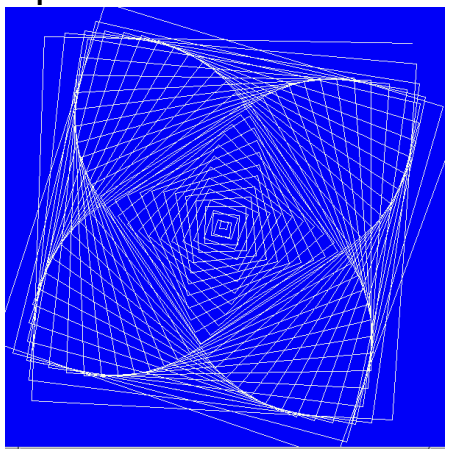
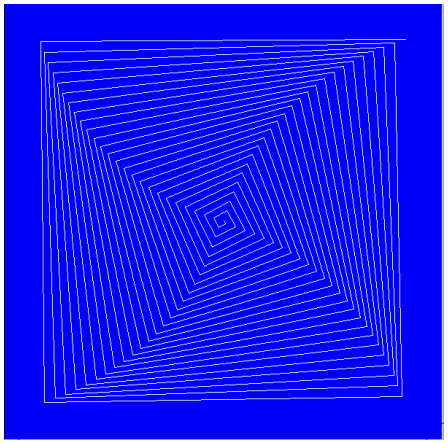

(If the lines in the picture vary in brightness that is a scaling effect.)

# Classroom use of the polypanel program 2

A good thing to do in the classroom is to discuss relationships between changes in the selected numbers and the pictures. E.g. what changes from the previous input would produce the picture on the right?



A teacher can invite suggestions as to how to produce effects like these:

In particular, what happens when the lines drawn have negative lengths?

How could you produce a smooth looking spiral curve?

The point is not just making pretty pictures, but understanding the relations between structure, process, and generic representations of classes of processes.

# How we can get machines to do things versus how machines can get themselves to do things.

The previous examples might, for some students, provide a starting point for asking questions like:

- If a robot or animal encounters different shapes in the environment how should it think about those shapes?
- One way of thinking about them is thinking about the processes that can create them
  (the philosophy behind the LOGO turtle, and our previous examples)?
- What other ways are there of thinking about shapes?
  e.g. using logical descriptions of parts and their relationships?
  or using pictures, sketches, diagrams, models?
- How do the different ways of thinking about spatial structure relate to different purposes for which we think about spatial structures?
- For what sorts of different purposes do we need to think about them?
- Why should a robot, or animal, ever need to think about how something looks as opposed to what its shape, orientation, and location are.
- How could such processes be replicated in future machines?

# Computation, Intelligence and the Study of Mind

A major area of impact of the development of computers has been to accelerate new thinking about the nature of minds – in humans, other animals and machines.

Although the label "Artificial Intelligence" (AI) suggests a focus on applications using machines, in fact most of the leading thinkers in AI, including John McCarthy, who invented them name, understand AI as much broader than the science and art of making smart machines.

See his online discussion:

`http://www-formal.stanford.edu/jmc/whatisai/node1.html`
"What is artificial intelligence?"

There are also relevant papers and discussions on the web site of another of the founders of AI, Marvin Minsky,

`http://web.media.mit.edu/~minsky/`

including the nearly final draft of his book, *The Emotion Machine*.

---

It can be illuminating to compare the the most sophisticated current robots with human toddlers and other animals.

Some relevant videos of children and robots are here
`http://www.cs.bham.ac.uk/research/projects/cogaff/movies/vid/`

# Varieties of learning

There is a lot of work being done on getting machines to learn many different sorts of things, e.g. learning to recognise objects, learning to use language, learning to work, learning to grasp things, and many more.

Most of that work is concerned with achieving successful learning, as part of the process of making a machine more useful.

However, as scientists rather than engineers, we may also want to model some of the incorrect, confused forms of learning that humans go through.

The next few slides illustrate an example, though there are many more examples in the developmental psychology literature.

Probably teachers have observed many examples of mis-learning that could be amenable to an analysis in terms of information-processing bugs.

Compare the BUGGY program developed by John Seely-Brown and colleagues, many years ago.

There are also very impressive robots that have expert behavioural competences, e.g. BigDog by Boston Dynamics, and the clothes folding robot by UC Berkeley:

- `http://www.bostondynamics.com/content/sec.php?section=BigDog`
- `http://www.willowgarage.com/blog/2011/06/06/solving-laundry-uc-berkeley`

Questions to be asked: how much do these (and other behaviourally impressive) robots know about what they are doing, what they have done, what they can do, what they did not do but might have done, why they did not do it, what could go wrong if they don't change their plans, etc.... ? Could a robot become a teacher?

# Reasoning quirks during development:
## reasoning about potentially colliding cars

Here is an example of a child who is nearly, but not quite, ready to reorganise and systematise some of the things he has learnt.

A child can learn to use a mixture of simulation and symbolic/verbal reasoning to draw conclusions and to justify conclusions.

That learning takes time and the process can go through buggy stages as illustrated by this example.


A racing car is on the left, and a van on the right.

The two vehicles start moving towards each other at the same time.

The racing car on the left moves much faster than the truck on the right.

Whereabouts will they meet – more to the left or to the right, or in the middle?

Where do you think a five year old will say they meet?

# One five year old's spatial reasoning


The two vehicles start moving towards each other at the same time.

The racing car on the left moves much faster than the truck on the right.

Whereabouts will they meet – more to the left or to the right, or in the middle?

One five year old was asked this question with two toy vehicles placed on opposite ends of a window-sill, facing each other.

**He answered by pointing to a location near 'b' !!!**

# One five year old's spatial reasoning


The two vehicles start moving towards each other at the same time.

The racing car on the left moves much faster than the truck on the right.

Whereabouts will they meet – more to the left or to the right, or in the middle?

One five year old answered by pointing to a location near 'b'
Me: Why?
Child: It's going faster so it will get there sooner.

What produces this answer:

- Missing knowledge?
- Inappropriate representations?
- Missing information-processing procedures?
- An inadequate information-processing architecture?
- Inappropriate control mechanisms in the architecture?
- A buggy neural mechanism for simulating objects moving at different speeds?

Would designing simulations for parts of this scenario, or the whole thing, help a child understand how to think about relative motion of this kind?

# Partly integrated competences in a five year old

Perhaps the child had not yet learnt that there can be constraints on ways in which items of information can be combined, like these:

- If two objects in a race start moving at the same time to the same target, the faster one will get there first
- Arriving earlier implies travelling for a shorter time.
- The shorter the time of travel, the shorter the distance traversed.
- So the racing car will travel a shorter distance!

The first premiss is a buggy generalisation: it does not allow for different kinds of 'race'.

The others have conditions of applicability that need to be checked.

Perhaps the child had not taken in the fact that the problem required the racing car and the truck to be travelling for the same length of time, or had not remembered to make use of that information.

Perhaps the child had the information (as could be tested by probing), but lacked the information-processing architecture required to make full and consistent use of it, and to control the derivation of consequences properly.

This is a simple example of a type of confusion I have often seen in students trying to think about processes: many features of a process are understood, but the learner has not developed a way of thinking about them simultaneously so as to ensure consistency. (Inadequate short-term memory mechanisms?)

(Compare being unable to see the impossibility of a Penrose triangle – or more complex Escher picture.)

# Discussing hypotheses before producing working models

Producing a full working model of the processes that produce the confusion about the car race would be very complex and difficult.

However, even for learners who have not yet got the experience and knowledge required to propose and implement a working model, it could nevertheless be useful to think about what the problems would be, on the basis of having done some much simpler programming tasks.

There's another type of development that can occur in learning about numbers or learning about shapes and shape interactions.

- The child learns some new concepts, or new types of behaviour
- The child constructs and plays with instances of the concepts or the behaviour type.
- As a result the child makes an empirical discovery – e.g. counting a row of objects left to right and counting the same set right to left produces the same object.
- Later the child acquires a deeper understanding and realises that those are not merely empirical discoveries: the truth of the generalisations can be derived by reasoning.

---

There are examples related to counting, measuring, ordering, and rearranging entities in,
J. Sauvy and S. Suavy, *The Child's Discovery of Space: From hopscotch to mazes – an introduction to intuitive topology*, Penguin Education, Harmondsworth, 1974,

A group with experience of simple AI programming could start discussing what forms of representation, what algorithms, what information-processing architecture could explain the observed phenomena, led by the teacher.

# How could biological evolution produce human or animal visual abilities?

**How could the ability to process information as humans and other animals do have been produced by evolution – apparently starting with molecules in a chemical soup?**

Some forms of human-like information processing are easier to model than others:

Some of the earliest AI successes were computer programs that learnt to play board games (e.g. checkers/draughts), worked out solutions to puzzles, made plans, and solved various problems.

There have also been various AI expert systems – some of which perform better than humans, e.g. at medical diagnosis or other tasks that depend on making good uses of much evidence.

But it has proved very much harder to give AI systems the ability to see and understanding processes in which moving 3-D objects interact, e.g. hands and cups, or an elephant's trunk and the things it picks up.

# Could mechanisms known to AI researchers, computer scientists, software engineers, etc., explain what human consciousness is?

There has been partial progress insofar as we have learnt over several decades to create processes in machines that have many of the features of mental processes, e.g.

- they cannot be observed and measured using physical devices
- they have semantic content (meaning) insofar as the use information and create and send information
- their interactions can produce physical effects (e.g. flight control systems, or spelling checkers that report things on a screen)
- in some cases the machine may be able to observe what it is doing internally and keep records of what it has done and improve its skills and knowledge as a result.

We have now learnt how to create running virtual machines in computers, with those competences, i.e. virtual machines that are not mere mathematical abstractions but can actually do things, e.g. landing an aeroplane in the dark.

Those virtual machines are not physical entities that you can see or measure by opening up a computer and looking inside or using voltmeters and other tools of physics.

For a taster of how to use ideas about the development of virtual machines as a product of biological evolution, see these slides:

`http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#darwin`
`http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#inside`

# Which children?

I have given a number of examples (and there are many more that could have been given.)

Which learners should be exposed to these problems and given the chance to start learning to build working models of individuals that perceive, think, deliberate and reason in deciding what to do?

Clearly the set of potential beneficiaries includes more than just the learners who wish to go into computing research or development, e.g. as software engineers.

The beneficiaries could include learners who wish to study a wide variety of different subjects e.g. in philosophy, biology, psychology, sociology, economics, education, neuroscience, psychiatry, etc.

Contrast

- Offering specialised versions for learners interested in biology, psychology, economics, linguistics, philosophy, mathematics.
- Offering a study of computation for modelling complex systems, including animal minds, as part of a general science syllabus.

# Towards a liberal education for the 21st Century

Some things should form part of a general educational system because they are worth knowing about in their own right for many different reasons, including all the main academic disciplines:

- physical sciences (physics, chemistry, astronomy, earth sciences, ...)
- biological sciences, including neuroscience, psychology,
- mathematics
- linguistics
- philosophy
- music and other art forms
- history, geography, literature, social sciences
- application fields: engineering, medicine, ....

A subset of worthwhile competences are specially important because they provide the tools for learning, thinking, reasoning and communicating about some, or all, of the others – it is normally assumed that these include the "three-Rs": Reading, wRiting, aRithmetic.

(Actually arithmetic is a small subset of mathematics, the real third competence.)

We need Five Rs, i.e. including programming broadly construed as learning to think about, analyse, explain and use tools for thinking about:

structures, processes and how structures can produce and be produced by processes.

Computers have been able to support such education for nearly 40 years – but alas the opportunity has been mostly ignored, and it is now very difficult to use it: but perhaps not impossible. **That's the assumption of this presentation.**

# Education – aims, and tools

## Some ideas about what education is for and how it can work: especially using opportunities provided by computing.

Successive well-meaning governments have got it badly wrong: especially in recent years in the UK!
(Both content and management – e.g. not allowing enough diversity in developmental trajectories.)

- School education, especially in early years, should primarily be about learning how the world (including ourselves) works, plus learning to think, learn, and communicate.

  Skills that serve those ends are deeply important – much more important (for children) than learning to use the latest gadgets – especially for children with high academic potential.

- Learning to use tools currently used in industry is no preparation for the real future.

  E.g. We should not teach physics by teaching children to drive tanks or fly airliners.

  **Good ways to think have much longer-lasting value.**

- Not much is known about the mechanisms and forms of representation involved in the processes through which humans learn and develop –

  but there are some hunches that look promising (educationally, if not politically – see below).

  We need better theories about how processes involving deep and effective learning can occur:
  what can happen in a learner's mind? E.g. How do ontologies develop?

- It is often assumed that the motivation that drives learning must always involve rewards – e.g.through teaching that provides fun and enjoyment.
- But evolution has produced, in humans (and some other animals), powerful learning mechanisms that do not need rewards: **what they really need is opportunities!**
- An idea from Vygotsky provides a useful concept for thinking about all this. (ZPD)
  (I use my own version of his idea: he can't be blamed for what follows.)

# A Learner's Zone of Proximal Development–ZPD

Educators need to understand and make good use of the learner's ZPD.
`http://en.wikipedia.org/wiki/Zone_of_proximal_development` (Originally Vygotsky's idea.)

Learning should include **meeting challenges:** if all challenges are too close to what the learner can do easily, there will be little learning (though perhaps much enjoyment).

If the challenge is too far beyond what the learner can do at all, there may be no learning.

In between those extremes is the learner's ZPD, with "near" and "far" boundaries:

- actions close to the "near" boundary accumulate information and make skills faster/more fluent. (How?)
- actions beyond the "far" boundary produce failure and frustration.
  (though sometimes useful seeds are planted for delayed learning effects).
- In between there can be piecemeal accumulation of information (factual, procedural, and conceptual): increments within a largely unchanging framework, including training cognitive reactions.
  (Think of examples: new phrases learnt, new kinds of process observed, new action combinations...)
- Closer to the "far" boundary of the ZPD, learning can be (temporarily) more disruptive, producing
  - awareness of not understanding,
  - a need for substantially new concepts,
  - pressure for a new type or layer of competence,
  - pressure for a new explanatory theory,
  - pressure for a new form of representation, with new modes of reasoning, predicting, explaining,
  - pressure for a new organisation for old material (e.g. with new abstractions) ...
  - new productive confusion, that eventually drives conceptual extension or reorganisation.

**How does this happen?** We don't yet have detailed working models – only hunches.

# At any age the ZPD can differ for different learners

Apart from genetic differences, children differ in many ways:

the kinds of toys, familiar actions, well understood household objects, social patterns, vocabulary and syntax, historical and political references, degrees and kinds of domestic harmony and disharmony...

all of which influence and limit the forms of learning available to the child at any time.

- New forms of learning often have to be based on familiar examples, and prior competences, so "the zone of proximal development" at each age will differ for different children for environmental as well as genetic reasons.
- Possible cognitive transitions at any age will differ between learners:

  the full set of possible routes to any major development **will not form a linear sequence** that all can follow, but a partially ordered network, through which different learners must take different routes.
- **The processes of teaching and assessment must allow for individual variation**

  including differences in both trajectories and speeds of traversal.
- This makes teaching very demanding – especially in large mixed-ability classes.
- The need to stretch all while discouraging none can make production of teaching materials tailored to the students very difficult.
- But humans and other animals can learn in environments that adjust themselves to the playing, learning, individual – e.g. like a part-built sandcastle, or meccano model.
- Computers can also help support personalised trajectories (e.g. using mini projects)
- However, only trivial things can be taught without generating (temporary) confusion.

# An important way in which computing can help

A partial solution is to provide opportunities for learners to select and follow their own routes – in a well designed space of possibilities for learning.

How can a learner become a good driver? By using suitable roads and vehicles!

- Divide a course into packages, where each package starts as simply as possible and then leads on to increasingly difficult challenges,
- Each package should include
  - some tasks everyone in the group can cope with
  - practice with varied sub-types of whatever is being learnt
  - some tasks all average students can cope with
  - clear specifications of goals all students should aim at, and indications of possible targets for high fliers (HFs).
  - some increasingly hard problems and challenges that can drive further deep learning in the HFs: learning to explore, hypothesise, test, revise, debug, reconsider, analyse, explain, document ...
- Some illustrations of these teaching ideas are presented here (needing improvement!)
  `http://www.cs.bham.ac.uk/research/projects/poplog/freepoplog.html#teaching`

# Supporting a variety of learners

Continued from previous slide.

- Make sure the High Fliers have opportunities to go on learning to their limit.

  Time will inevitably be a constraint: they also have to learn when to move to another package.

- Tools should give teachers great flexibility:

  Packages may need to be re-ordered, or broken up into smaller packages for younger learners, or disadvantaged learners.

  Teachers need to be able easily to change the examples to suit their context.

- Some illustrations of these teaching ideas are presented here (needing improvement!)
  `http://www.cs.bham.ac.uk/research/projects/poplog/freepoplog.html#teaching`

# The Three Main Points

## 1. Cognitive learning and development involve growing and populating an **information-processing architecture.**

Driven to a large extent by playful exploration – including things going wrong

Bugs/errors/mistakes are not bad things to be avoided: they are a deep resource for learners.

(Compare ideas by John Holt, Ivan Illich, Alan Kay, Seymour Papert, Gerald Sussman, Mitchell Resnick, Marvin Minsky, Jean Piaget, and others. Also philosophers of science: K. Popper revised by I. Lakatos)

## 2. Computers can provide powerful provocations and support for the process e.g. playing with AI programs and AI programming concepts.

Scratch, Alice, Netlogo, Roblox and other new graphics-based tools are powerful engagers for young learners – but can be contrasted with simpler, smaller, and more flexible and extendable learning environments, some more focused on AI and some less graphics-based.

We need especially to understand the importance of textual manipulation to support abstraction and recombination of ideas. Some examples of what can be done are here:

`http://www.cs.bham.ac.uk/research/projects/poplog/examples/`

## 3. One possible way to get more of this into schools.

Make remotely-usable systems available remotely (e.g. using Xming and SSH clients on Windows PCs) to connect to well-managed, shared linux machines, supporting strong collaborative learning and teaching, especially introducing various types of AI programming.

Also Tim Bateson has made a very powerful resource available that can be run on Windows and other platforms using VirtualBox. See

`http://www.cs.bham.ac.uk/research/projects/poplog/ova/A-README.html`

# Example: A challenge for older learners

Suppose a computer could not do anything with numbers but could manipulate symbols.
Could you get it to use lists of symbols to represent numbers, and operations on lists to represent numerical operations?

E.g. suppose the first few numbers are

[]

[x]

[x x ]

[x x x]

[x x x x]

It is relatively easy(?) to define list-processing algorithms to represent addition, subtraction (of a smaller number from a larger one) and multiplication.

How would subtraction of 5 from 3 be dealt with?
How should negative numbers be represented? What about division?

See some suggestions here (using Pop-11):

`http://www.cs.bham.ac.uk/research/projects/poplog/teach/teachnums`

(An introduction to “unary arithmetic”.)

(Can **you** explain why multiplying two negative numbers should give a positive number?)

# Why nearly everything?

There are many types of explanation, e.g. using physics, chemistry, ...

It is widely believed that the science of information processing systems can all be reduced to physics and chemistry, along with appropriate mathematics.

However, there are interacting patterns of structure and process which are "implemented" using physics and chemistry, but whose description requires going beyond the language of physics and chemistry. Examples include:

- machines playing games (e.g. "winning", "threatening" are not concepts of physics);
- machines considering and trying to choose between different goals ("goal" is not a concept of physics or chemistry);
- social phenomena, such as voting, persuasion, deception, education, envy, theft, murder, punishment, and many more – all involve physical mechanisms, but describing the patterns of interaction needs more than the language of physical sciences.
- All involve "virtual machinery": `http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#talk76`

There are far more types of chemical compound than types of individual atom. (Why?)

There are far more types of chemical process than types of physical process that include no chemical changes. (Why?)

Conjecture:

There are far more possible patterns at biological and social levels of organisation, composed of physical and chemical processes, involving information processing, than types of physical and chemical process that do not involve information processing.

I.e. there are more types of information-processing virtual machines than types of physical machine.

# Toolkits for thinky social agents

There are various packages that have some support for such interacting simulated agents with “thinky” capabilities, e.g. Netlogo (though most of the examples provided are more “bumpy” than “thinky”).

`http://ccl.northwestern.edu/netlogo/`

However, supporting simulation of motion of agents with articulated bodies, e.g. with jointed arms and hands that can be used for manipulating objects in the environment, is much more difficult than supporting motion of rigid, relatively unstructured objects that can merely move, rotate, bump, push, etc.

The toolkits vary in the degree of sophistication they allow within each agent.

For more ambitious “thinky” projects without complex physical interactions, our SimAgent toolkit may be useful

`http://www.cs.bham.ac.uk/research/projects/poplog/packages/simagent.html`

`http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#simagent`

The toolkit is included in this package

`http://www.cs.bham.ac.uk/research/projects/poplog/ova/A-README.html`

There are many new toolkits being developed by researchers in AI and Robotics many of which will work with simulated agents in simulated physical environments, avoiding the cost and and time on low-level problems that inevitably arise in connection with physical robots – though dealing with those problems would suit some students well.

Ideally, some learners should have access to tools for relatively easy development of “thinky” agents (with rich cognitive and motivational capabilities), and mechanisms for adding greater sophistication (e.g. more concurrent, asynchronous, processes within each agent.)

# Not just computer-based tools

NOTE: Computers make powerful new forms of learning possible, but other things must not be forgotten ...

A major part of my education was playing with meccano out of school hours between the ages of about 5 and 10, over 60 years ago.
Not everything that is important in human learning can be done with computers:
The 3-D world can be a powerful teacher: it drove much of our evolution.

Also pencil, paper, string, elastic bands, pins, buttons, .... as illustrated in:

**The Child's Discovery of Space: From hopscotch to mazes – an introduction to intuitive topology**

Smaller J. Sauvy and S. Suavy,
Penguin Education,
Harmondsworth, 1974,
Translated from the French by Pam Wells,

Creative teachers could invent programming tasks based on some of the examples there.

Additional inspiration can come from the work of John Holt (e.g. How Children Learn)

And a little book by a great mathematician: G. Polya How To Solve It.

There's lots more!

# The CAS Mission Statement

This is the mission statement for Computing At School, composed in discussions at the CAS working group meeting 27-28th April 2010, at Microsoft Cambridge.

**MISSION STATEMENT:**

Computing has transformed the way we communicate, work and play in recent decades.

However, it is not obvious to most people that it is a broad, deep, exciting, and fertile discipline.

Its full potential can only be realised through great creativity and ingenuity in working with concepts, theories and challenges.

It has also been a powerful source of new ways of thinking that have influenced many other fields, as diverse as biology, mathematics and philosophy.

CAS aims to support and promote this vision of computing in UK schools, by providing opportunities and resources to help all educators and learners to better understand, use and develop computing in a variety of activities and fields.

This document is written as a partial elaboration of the above.

I have attempted to assemble some examples of ways of teaching computing and programming, not as a collection of “useful skills” but as a way of expanding our ability to understand other things, including human minds, animal cognition, and their evolution.

# The End – For Now