COMPUTING CONCEPTS AND MODELS TO ENRICH THE ENTIRE SCHOOL CURRICULUM

***The Turtle System* has recently been significantly developed at Oxford University thanks to a project co-funded by the Department for Education. Peter Millican, Professor at Hertford College and author of the system, has written a free book and suite of programs illustrating the value of CS concepts to many subjects.**

FREE RESOURCES FOR TEACHERS

It is well known that Computational Thinking is widely applicable, and that programs can be used to illustrate and explore many varied phenomena, especially in the sciences. But most students find it very hard to write such programs for themselves, and even expert teachers typically lack the time to develop more than one or two.

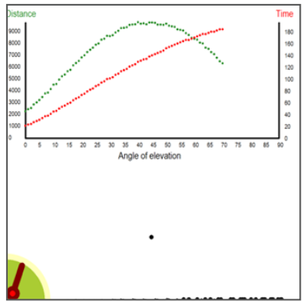
'*Computer Science Across the Curriculum*' – a book to be distributed free to secondary schools – addresses this problem, and illustrates the implementation of computer models in many different disciplines.

The first chapter explains how to get started with the *Turtle System*, and its basic concepts, after which the second provides a simple introduction (in BASIC, Pascal, and Python) to animation, modelling of motion, and user input by keyboard or mouse. Then follow chapters on Physics, Cellular Automata, Chemistry, Biology, Mathematics (from chaos, recursion and self-similarity to waves), Computer Science (including game algorithms), and Philosophy and the Social Sciences (e.g. models of co-operation and segregation).

The material from the science chapters, illustrated in this article, can easily be incorporated into lessons even without the *Turtle System*, as most programs can run direct from www.turtle.ox.ac.uk/csac. Select one, and it will be loaded into *Turtle Online*; then clicking 'RUN' will execute it in your web browser.

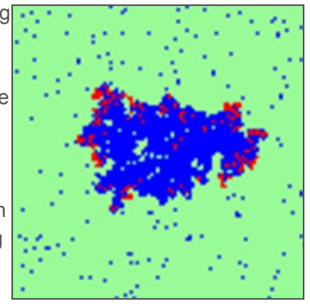
The final chapters — on Philosophy and Social Sciences — are less likely to be relevant to conventional curricula, but are designed to inspire students and teachers to appreciate how computer-based methods open new and exciting possibilities for these disciplines.

Physics provides some obvious topics for modelling. The simple graphical interface of *Turtle System*, based on canvas *x* and *y* coordinates, makes it very easy to model motion in two dimensions using a procedure that moves the turtle repeatedly by the appropriate *x* and *y* velocities, drawing an animated projectile as it goes. The automated cannon program shown here (with smashed balls littering the ground and one in flight), gradually raises the cannon from 0° towards 90°, building up graphs to show how flight time and distance vary with the initial angle. Students thus gain a deeper, practical appreciation of the relevant theory, from the trigonometric calculation of initial x and y velocities, to the application of gravitational acceleration to the *y* velocity.



Another more challenging program models firing a rocket into orbit, using real physical values and tracking the rocket's position to the nearest metre each second, as well as its velocity (and acceleration) to the nearest millimetre per second (squared). With control over only the initial thrust, time of thrust, and firing angle, it is surprisingly difficult to achieve orbit, but students are encouraged to develop the program further, taking account of other physical factors such as weight loss (as fuel is burned) and the Earth's rotational velocity. Again, there is scope for entertainment, exploration, and substantial learning beyond the syllabus.

Cellular automata offer lots of interesting possibilities, starting with a simple version of a standard model of epidemics (pictured) which can be used to illustrate the value of inoculation. Then, a simple implementation of the Game of Life – just thirty lines long – aims to motivate learning about binary numbers, Boolean operators and encoding methods, using hexadecimal numbers to store information as coloured pixels.



Another program illustrates Wolfram's theory of one-dimensional automata, generating patterns that are strikingly reminiscent of some found in natural shells (such as *Conux textile* shown).

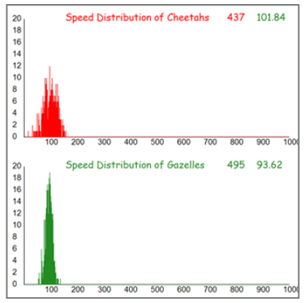
The chapter on Chemistry uses similar techniques to model diffusion of liquids before moving on to explain and apply simple atomic theory, starting from Brownian motion (which also gives an opportunity to illus-

Turtle System is based on Turtle Graphics, an idea invented by Seymour Papert. This sort of programming, and the results it produces, are easy to understand because they are so immediately visual. But the Turtle System discussed here shows that Papert's idea can go well beyond simple graphics, to provide a basis for fascinating and powerful programs that can explore many cross-curricular concepts through a variety of computer models.

CS4FN: FROM FASCINATION TO IMPLEMENTATION

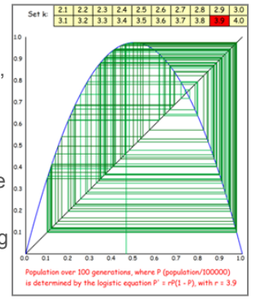
This new book was developed in collaboration with *CS4FN*, the popular magazine (and website www.cs4fn.org) based at Queen Mary University of London. Most students (and even teachers) are likely to find it hard to translate general ideas into precise algorithms, but these programs illustrate how to do so, and are sufficiently short to be relatively easy to understand, modify and learn from. Topics connected with CS4FN include animal behaviour (e.g. ant trails), animation, artificial intelligence (e.g. playing Nim or Noughts and Crosses), cellular automata (e.g. Game of Life and morphogenesis), chaotic phenomena, disease epidemics, evolutionary models, fractal art, graphics and image encoding, logic, mechanics, Prisoner's Dilemma, and searching.

trate more Physics, in the impact of particles). Again independent exploration and interdisciplinary crossover are encouraged, for example by inviting students to combine the diffusion and epidemic models to demonstrate how mobility of infected individuals can radically alter the dynamics of disease spread.

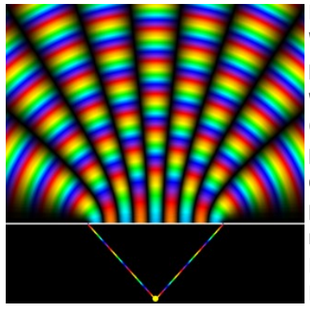


Biology also provides many other excellent examples, from the competitive evolution of ever-faster cheetahs and gazelles, to evolutionary fixing of the sex ratio, to modelling of coordinated movement such as trail-following in ants or flocking of birds. Yet again there is plenty of scope for students to take these ideas further, with the necessary programming techniques explained and illustrated to enable these models to be developed through, for instance, the introduction of obstacles, rival species, and predators.

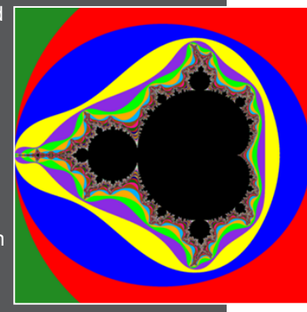
Another biological model, of insect population, based on the well-known *logistic equation*, provides an excellent illustration of *chaos*, a form of non-linear dynamics whose wide applicability to many different domains has come to light precisely through the exploration of computer models. For more on this fascinating topic, see the sidebar.



Next comes a chapter on wave interference, including illustrations of Fourier decomposition (using Hugh Wallis's impressive 'wave superposer' program) and a model of wave patterns within Young's two slit experiment (shown). Though quite advanced, this provides illustrative material for teachers to supplement their classroom explanations, and will help students who may be reading books such as Richard Feynman's *QED* or Brian Cox and Jeff Forshaw's *The Quantum Universe*.



THE MATHEMATICS OF CHAOS



The most celebrated icon of chaos is the Mandelbrot set, and this book explains exactly what that is, providing a simple program that generates a picture of the complete set (shown right), as well as allowing more detailed 'zooming in' to regions within it. The aim here is to give real understanding of the relevant theory by providing genuine implementations – in simple computer language – which can be examined, run and modified by students themselves. Those who have read about chaos, without ever learning exactly what it is, will thus be enabled to dig much deeper.

The chaos chapter of *Computer Science Across the Curriculum* then moves through discussion of the well-known Sierpinsky triangle, which can be generated in several ways, some of these involving *iterated function systems* similar in principle to the Mandelbrot process. It is fascinating to discover how the simple specification of iterated functions, by resetting a few parameters, can generate – from what is essentially the very same program – such different patterns as Michael Barnsley's famous fern and the dragon curve (below).

These patterns are known as 'fractals', displaying self-similarity in the complete pattern and their sub-parts, so that, like the Mandelbrot set, in principle they have infinite detail 'all the way down'.