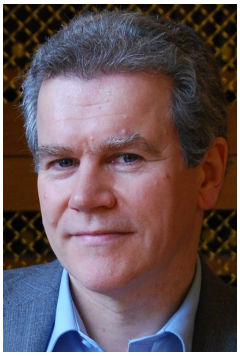


# USING THE TURTLE SYSTEM: AN EASY WAY TO START TEXT-BASED PROGRAMMING

***The Turtle System*, with teaching resources and coursework setting and marking tools, is available free thanks to a new project at Oxford University co-funded by the Department for Education. Peter Millican, Professor at Hertford College, Oxford, shows how to get started, both using the system and teaching with it.**

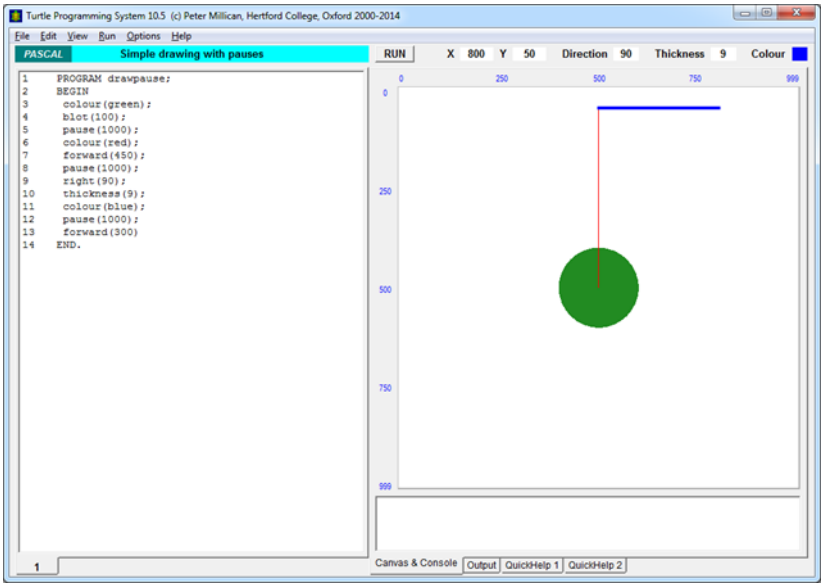
## GETTING STARTED: HOW TO DOWNLOAD AND INSTALL

In the last issue of SWITCHEDON, I explained the principles behind *The Turtle System*, including its support of multiple "barebones" languages, simple but powerful graphics facilities, ease of setting up, precisely targeted error messages, and links to Computer Science through its use of a virtual *Turtle Machine* whose memory and machine code can be inspected in detail (and which enables *Turtle* applications to be run on the web and mobile devices as well as PCs). If any readers are interested in contributing to this project (either paid or unpaid), by helping to design follow-on course materials (which might be in specialist areas), please do let me know. I can be contacted at the address peter.millican@hertford.ox.ac.uk

To get the latest version, browse to www.turtle.ox.ac.uk and click on the "Download" button. If warned that it is potentially unsafe, do not worry: this is just because your computer doesn't recognise the program, but rest assured that nothing on this Oxford University site is malicious. Having downloaded the file ("TurtleSystem.exe"), you can put this on your desktop or within any folder on your computer or network, and run it from there. *The software will make no further changes to your computer system or network.*

Download the Turtle System
Windows Only

Having started the program, click on the "Help" menu; then within the "Examples 1" submenu, select "Simple drawing with pauses". This will load a short program in the system's Editor (at the left), and if you now click on the "RUN" button (top middle), you will see a picture being drawn on the Canvas (at the right) as shown below.




It's worth running this program several times to note exactly what it is doing and how the screen changes to reflect this. The invisible "turtle" starts in the middle of the 1000×1000 Canvas (where the X and Y coordinates are both 500), pointing North (i.e. direction 0º), and as it moves around in response to the commands "forward(450)", "right(90)" and "forward(300)", its position and direction are shown in the boxes at the top-right. Between movements, three "pause(1000)" commands tell it to pause for 1000 milliseconds each time, while three "colour" and one "thickness" commands tell it what kind of line to draw as it moves. The current Thickness and Colour settings are shown (in real time) in the boxes at the top-right, next to the "X", "Y", and "Direction" boxes. In this screenshot, we see the Pascal version of the program, which is currently best supported and particularly suitable for beginners, but there is also a Java version (available through "Languages" after selecting the "Power User Menu" option), with BASIC and Python also planned soon. Thus pupils can easily prepare for moving onto a variety of other systems.

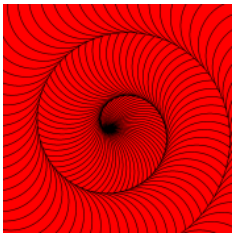
WebForum

TurtleOnline

This can be accessed at www.turtle.ox.ac.uk, where full documentation and a guided tour are available. Teachers can upload course materials, invite pupils to register for relevant courses, and have work submitted and collated online. Submitted *Turtle* programs can be viewed and run in a web browser, making assessment as convenient as possible. Moreover all this is available without having to prepare your own materials, since we have already commissioned ready-made courses from expert teachers, complete with PowerPoints, example programs, help resources, lesson plans, schemes of work, assessment criteria and targets, to provide all you need in the classroom and carefully tailored to "tick the boxes" on the new National Curriculum, as well as being engaging and educationally fulfilling.

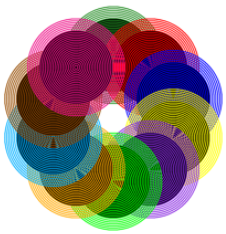
*Turtle System is based on Turtle Graphics, an idea invented by Seymour Papert. This sort of programming, and the results it produces, are easy to understand because they are so immediately visual. But the Turtle System provided here shows that Papert's idea can go well beyond simple graphics, to provide a basis for fascinating and powerful programs that introduce fundamental concepts of software engineering and artificial intelligence.*

```
PROGRAM forloop;
VAR count: integer;
BEGIN
 for count:=1 to 200 do
  begin
   forward(count/3);
   right(5);
   colour(red);
   blot(200);
   colour(black);
   circle(200)
  end
END.
```



**Example 1**

```
PROGRAM nestedloops;
VAR countblot: integer;
    countcirc: integer;
BEGIN
 penup;
 for countblot:=1 to 10 do
  begin
   forward(260);
   colour(black);
   blot(150);
   colour(countblot);
   for countcirc:=1 to 25 do
    circle(countcirc*8);
   back(260);
   right(36)
  end
END.
```

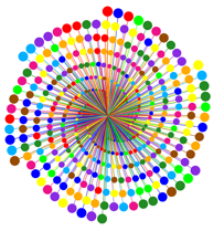


**Example 2**

```
PROGRAM parameterproc;
VAR count: integer;

  Procedure prong(len: integer);
  Begin
   forward(len);
   blot(len/20);
   back(len)
  End;

BEGIN
 for count:=360 downto 1 do
  begin
   randcol(10);
   prong(count+100);
   right(61)
  end
END.
```



**Example 3**

# LEARNING FROM EXAMPLES

A very easy way for pupils (and teachers) to get used to the Turtle System and its possibilities is to play with some of the built-in example programs, running and editing them to see what happens. These start from straightforward "turtle graphics" drawing, then quickly bring in "for" loops to show how easy it is to create striking effects even with very short programs (see Example 1 above). Before long we meet nested loops (Example 2), then simple procedures (Example 3).

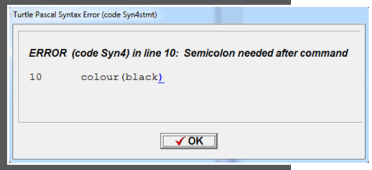
These are followed by examples illustrating all the main programming structures and commands, including "repeat" and "while" loops, recursive patterns, text printing and colour effects, keyboard and mouse input, mathematical and string functions, animation and simulation of physical systems, and various applications including video and strategy games, and cellular automaton models (e.g. the Game of Life).

| Expression | Level | Count | Program Lines | Category |
|---|---|---|---|---|
| | | | | *Turtle: relative movement* |
| back | 1 | 1 | 8 | |
| forward | 1 | 1 | 6 | |
| right | 1 | 1 | 16 | |
| | TOTAL: | 3 | | |
| | | | | *Turtle: drawing shapes* |
| blot | 1 | 1 | 7 | |
| | TOTAL: | 1 | | |
| | | | | *Other Turtle commands* |
| randcol | 1 | 1 | 14 | |
| | TOTAL: | 1 | | |
| | | | | *Command structures* |
| for | 1 | 1 | 12 | |
| procedure | 2 | 1 | 4 | |
| | TOTAL: | 2 | | |
| | | | | *Subroutine calls* |
| prong | 0 | 1 | 15 | |
| | TOTAL: | 1 | | |

***The automatic program analysis for Example 3 above***

## SUPPORT FOR LEARNERS

As noted earlier, the *System*'s error messages are very precisely targeted, wherever possible giving a clear and specific instruction for remedying the error (as shown here) so that pupils can experiment and learn effectively without requiring constant supervision. Convenient illustrations and explanations of program structures or commands are provided through the two "QuickHelp" tabs beneath the Canvas, with "QuickHelp 1" giving 8 summary pages on the main features, while "QuickHelp 2" provides listings of the *Turtle* commands available. These are organised by level of difficulty (with just the simple commands shown initially) and by functional category (so users of any level can quickly identify needed functions). The "Edit" menu provides an "Auto-format" facility which neatly indents programs to a standard pattern, encouraging good practice and enabling "scope" errors to be identified easily.




## SUPPORT FOR TEACHERS

The "Auto-format" also, of course, makes programs easier to mark. Moreover the system can easily be set up (through the "Power User Menu" option) to auto-format and/or run a program as soon as it is loaded – in which case examining a pupil's program requires no more than clicking on the relevant file. Another helpful feature is the "Usage" tab, which counts and lists the commands used in a program, organised by category (pictured left is the analysis of the "parameterproc" program—Example 3). I put these facilities into the original system when using it to introduce programming to large groups of "elective" students at Leeds University. By designing assessments accordingly (e.g. "write a program of at least 50 lines, using at least two looping structures …, to produce … effect"), it became possible to assess students' work extremely efficiently, freeing up a huge amount of time for face-to-face teaching rather than tedious marking.