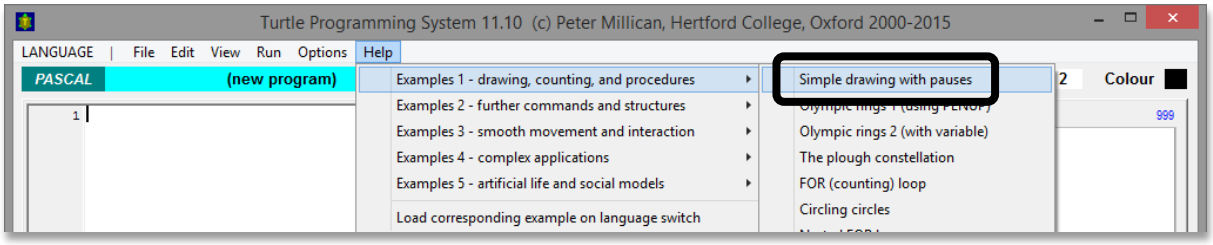
Self-Teach Exercises: Getting Started

Turtle Pascal

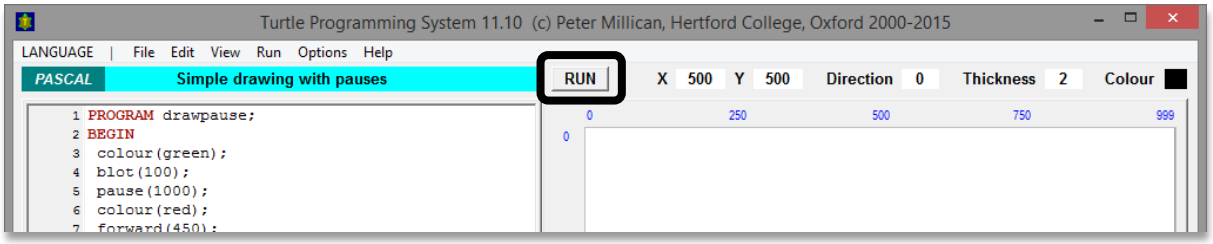
0.1 Select "Simple drawing with pauses"

Click on the "Help" menu, point to "Examples 1 – drawing, counting, and procedures", and select the first program on the list ("Simple drawing with pauses").



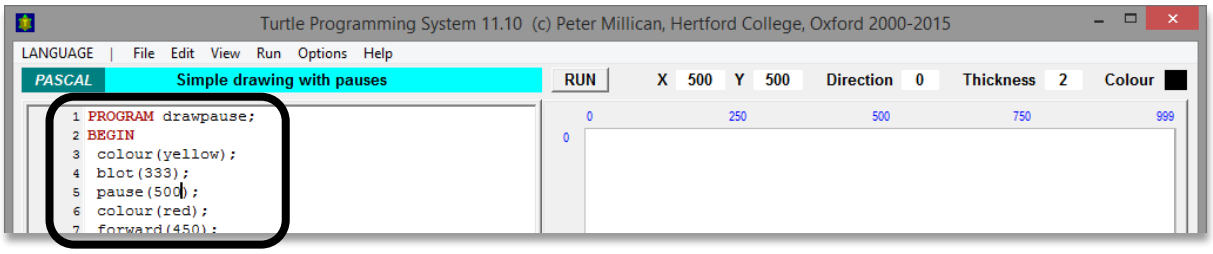
0.2. Click on RUN – then click it again!

When the program has loaded, click on the **RUN** button and see what happens. Then do it again… Can you work out how the commands are having the effects that you see on the Canvas?



0.3. Try editing the program

Make some simple changes to the program, like changing the colours, blot sizes, distances, angles, or pause times. Then **RUN** the program again, and repeat…



0.4. Now read about the program

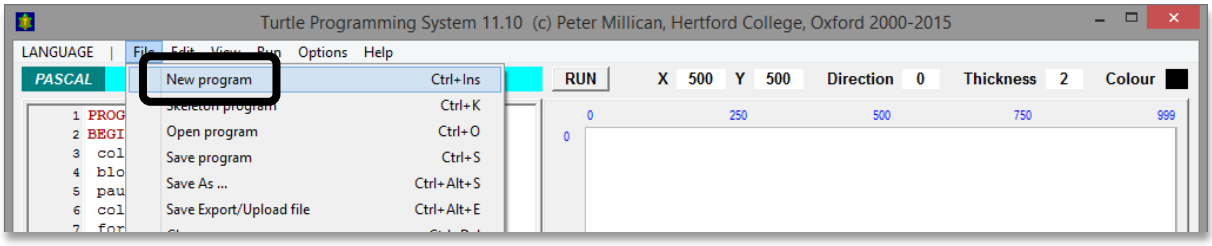
At the end of this handout, you'll find a section called "The Program". Read this through, and ask if there's anything you don't understand.

0.5. Familiarise yourself with the built-in help

You've already seen that the "Help" menu contains quite a lot of example programs for you to examine and play with. Take a quick look also at the "QuickHelp 1" and "Quickhelp 2" tabs at the bottom of the screen. "QuickHelp 1" gives a very brief summary of all the main programming structures (which you will be learning soon, so don't worry about those now). "QuickHelp 2" summarises all the built-in commands (such as `blot`, `colour`, and `forward`), and also shows the standard colours, fonts, and cursors available.

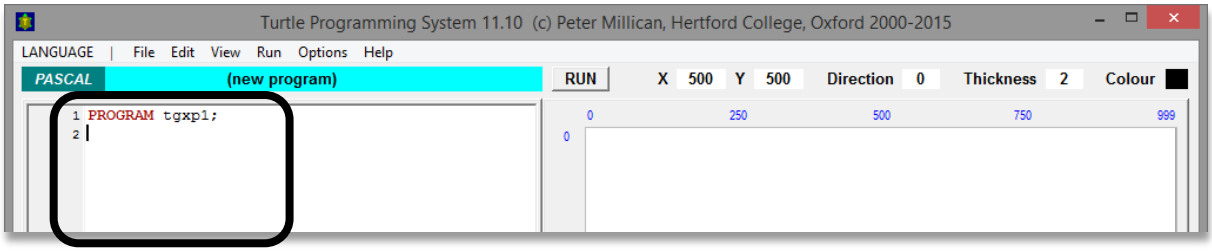
1.1. Clear the program

Click on the "File" menu, and then on "New program". ("Skeleton program" gives you a very short program with **PROGRAM**, **BEGIN**, and **END.**, and some example commands already written in.)



1.2. Start typing in your own program

You'll see the number "1". Click just after it, and type PROGRAM `tgpx1;`. Then press the ENTER key—this should move you to line 2.

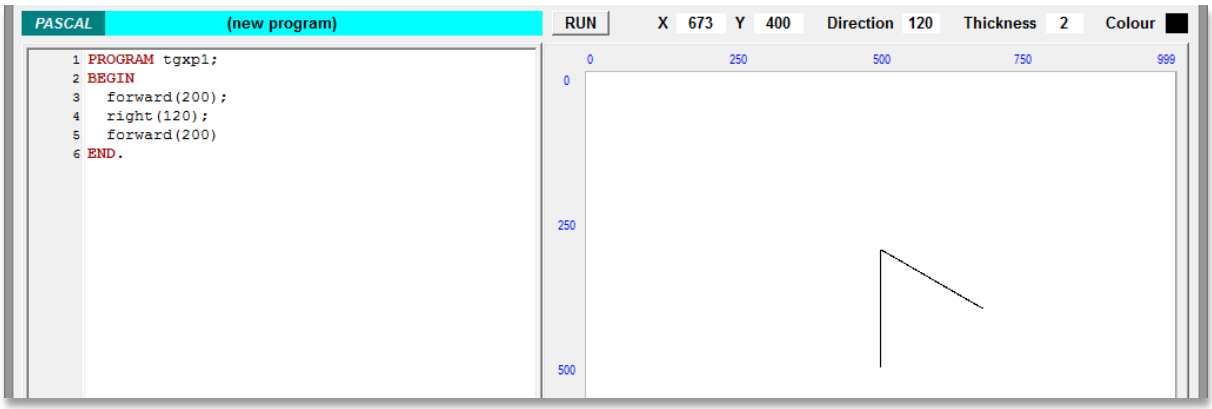


Carry on typing the rest of the program, pressing ENTER every time you need to move to a new line. Your program should look like this:

```
PROGRAM tgpx1;
BEGIN
  forward(200);
  right(120);
  forward(200)
END.
```

1.3. RUN the program, and see what it does

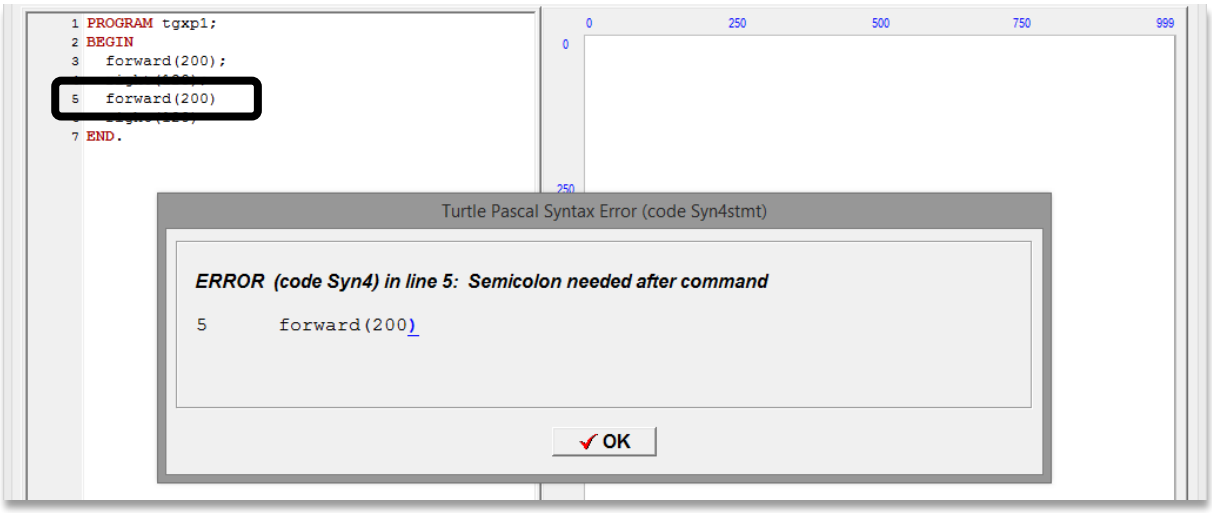
On the Canvas on the right, you should see two sides of an equilateral triangle. Notice that the current position of the Turtle is shown by the **X** and **Y** boxes. **X** gives the distance from the left of the Canvas, and **Y** gives the distance from the top. The Canvas is 1000 units square, so **X**=500 and **Y**=500 is in the centre. **Direction** gives the direction (in angles) that the Turtle is facing, with 0 meaning North, 90 meaning East, 180 meaning South, and 270 meaning West.



1.4. Complete the triangle

Now your first task is to add new commands to the program so that it draws a complete equilateral triangle. Can you see how to do this?

Note, by the way, that whenever you have one command followed by another, the first command must be followed by a semicolon ";". If you get this wrong, the system will give you an error message, highlighting the exact point where the error has been found.



You do not need to put a semicolon after the last command of your program (before the "END."), but you can put one there if you like—extra semicolons at the end of commands do no harm.

1.5. Straighten the triangle

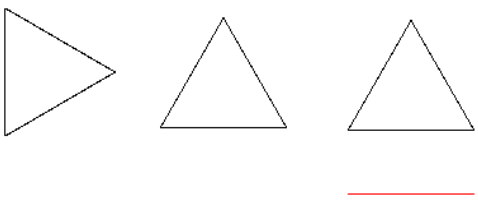
The complete triangle should look like image 1 below. Can you see how—by adding just one line to your program—you can make it look like image 2?

1.6. Add a red line underneath

Next you need to draw a red line underneath the triangle, 100 units below it (see image 3). To achieve this, you will have to think about a number of things:

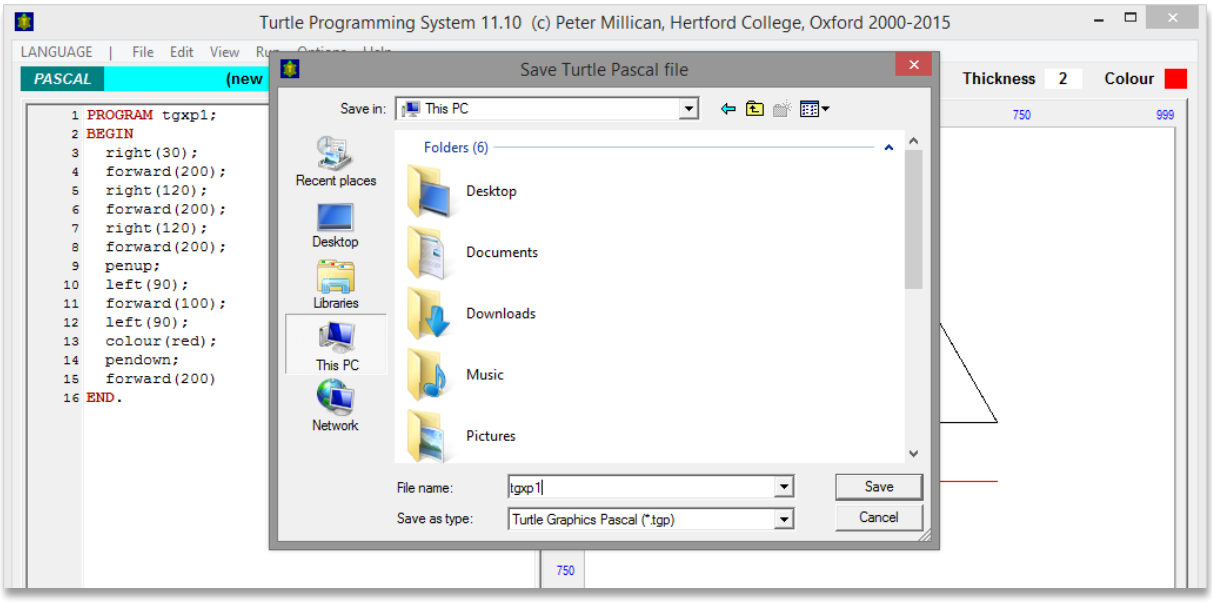
a) Where is the Turtle when it finishes drawing the triangle (hint: draw a small blot there if you're not sure), and which direction is it facing?
b) What commands are needed to get it from there to the place where the red line starts? (Hint: you can use `right(…)` or `left(…)` to turn the Turtle, and `forward(…)` or `back(…)` to move it.)
c) How can you get it there without drawing a line that you don't want? (Hint: use the `penup` command to make the Turtle take its pen up from the Canvas, so that it moves without drawing, and use `pendown` to make it start drawing again.)
d) How do you draw the red line? (Hint: `colour(red)` changes the colour.)

1 2 3



1.7. Save your program

If you want (and if you are using a computer where you have the appropriate permissions), you can now save your file. Click on the "File" menu, then on "Save As …". When you're asked to give a file name, type "tgpx1" or "TGPX1" (it doesn't matter whether you use capital or small letters). Then click on "Save".



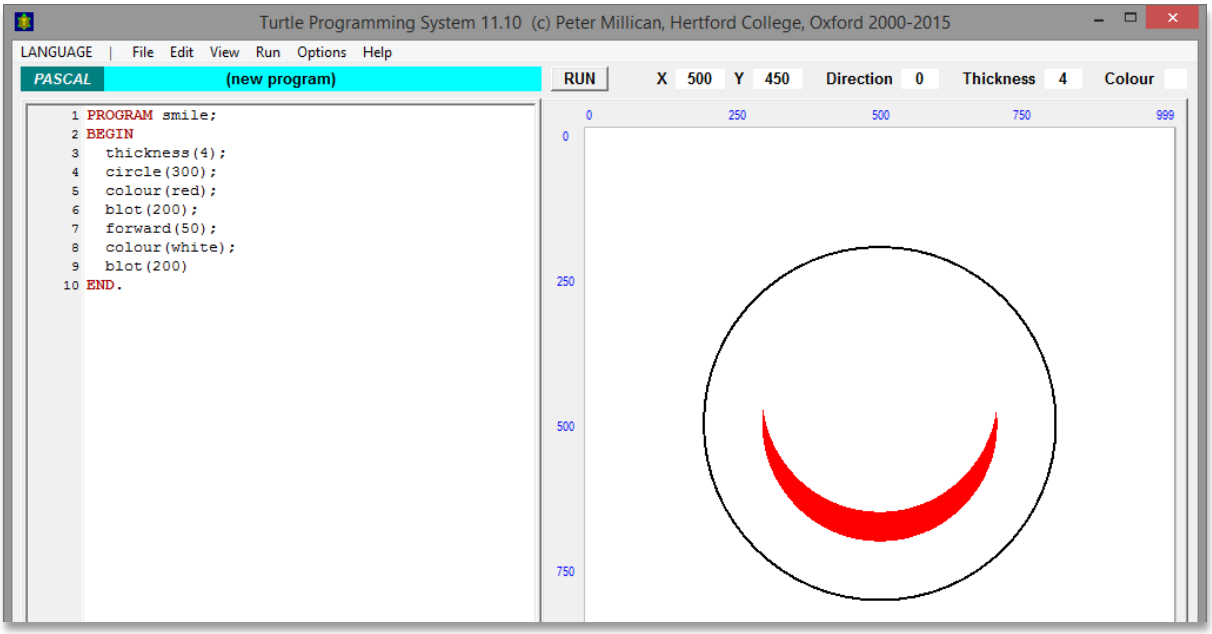
That's the first exercise done—you've written, and saved, your first *Turtle Pascal* computer program!

2.1. Clear the program

As before, click on the "File" menu, and then on "New program".

2.2. Follow the Exercise 2 instructions

Now see what you can do with Exercise 2, which asks you to draw a smiley face. For a clue on how to do a smile very easily, look at the following screenshot, showing a program which draws a white blot over the top of a red blot, but with a shifted centre:



The Program

The program is a sequence of commands in the programming language *Turtle Pascal*, which you type into the Programming Area on the left of the screen, and which are executed when you click the **RUN** button. These commands determine what is drawn on the Canvas (the square area on the right of the screen). You might find it helpful to think of them as instructions being given to a small (and invisible) *Turtle* which moves around the Canvas, leaving a coloured trail and drawing shapes as it goes.

The easiest way to see how all this works is to go to the "Help" menu at the top of the screen, select "Examples 1 - drawing, counting, and procedures" and click on the first of these, called "Simple drawing with pauses". This will load the following program into the Programming Area:

```
PROGRAM drawpause;
BEGIN
  colour(green);
  blot(100);
  pause(1000);
  colour(red);
  forward(450);
  pause(1000);
  right(90);
  thickness(9);
  colour(blue);
  pause(1000);
  forward(300)
END.
```

If you now click on **RUN** you will see the effect of this program on the Canvas. Here is a brief explanation, which should be sufficient to enable you to write similar programs of your own:

Every program has to have a one-word name (here `drawpause`), which is given after the word **PROGRAM** and followed by a semicolon; then the commands of the program itself are "bracketed" between **BEGIN** and **END** and separated from each other by semicolons.[1] The final **END** must be followed by a full-stop, which signifies the very end of the program. In this example the first command is `colour(green)`, which specifies the drawing colour as green; then `blot(100)` accordingly draws a green "blot" (i.e. a filled-in circle) of radius 100 around the initial position of the Turtle in the centre of the Canvas.

Next we have a `pause(1000)` command, which tells the Turtle to pause for 1000 milliseconds (i.e. 1 second). Then after another `colour` command we have `forward(450)`, which instructs the Turtle to move forward from its initial position by 450 units. This will leave a red trail, in accordance with the previous `colour(red)` command. After another `pause(1000)`, the Turtle is told to turn right by 90 degrees with `right(90)`, then `thickness(9)` tells it to thicken its pen from the standard 2 unit thickness to 9 units. This means that after `colour(blue)` and another `pause(1000)`, the final command `forward(300)` will draw a thick blue line of 300 units in what is now the direction of the Turtle (i.e. horizontal, having turned right by 90 degrees from its original vertical direction).

[1] Note that capitalization in *Turtle Pascal* is entirely optional – **PROGRAM**, **BEGIN**, and **END** are shown in uppercase only for emphasis, and it would make no difference if you changed them to lowercase and/or capitalised other words.