MicroWorlds and Mathematics: Build a (Turtle) House
John Gough — Deakin University — jugh@deakin.edu.au


It is well known that Logo turtles can draw houses, made from a square and a triangle. (Regrettably this is sometimes the only thing that some people think Logo programming is able to do. Fortunately we know better.) Here is a different turtle house. Notice the way several small procedures are used to build the whole house. This modularised approach to programming is one of Logo's most powerful features. Such a larger procedure which draws on or "calls" other procedures as command words, is a referred to as a "super-procedure".

to house
  wall1
  wall2
  doormat
  door
  window1
  window2
  roof
end

to wall1
  pd repeat 2 [forward 40 rt 60 fd 100 rt 120]
end

to wall2
  pd repeat 2 [forward 40 lt 60 fd 60 lt 120]
end

to doormat
  pu rt 60 fd 20
  pd repeat 2 [fd 40 rt 60 fd 20 rt 120]
  pu bk 20 lt 60
end

to door
  pu rt 60 fd 30 lt 60
  pd repeat 2 [fd 30 rt 60 fd 20 rt 120]
  pu rt 60 bk 30 lt 60
end

to window1
  pu rt 60 fd 60 lt 60 fd 10
  pd repeat 2 [fd 20 rt 60 fd 20 rt 120]
Strictly speaking, this contains more turtle turns and moves than is needed. But I have made sure that each sub-procedure finishes by returning the turtle to its initial starting point, and also leaves the turtle heading in the same direction it started from.

One of the benefits of this, is that when the next procedure is used, we know exactly where the turtle starts from, and its initial orientation. Another benefit is that we could, if we liked, swap any of the sub-procedures around, and the drawing of the house would still work the same. This would be different if the whole superprocedure depended on each sub-procedure being used in some correct order, but would not work if the order were (perhaps accidentally) altered.

We can use our procedure to create a street of houses.
We could also draw our house in any orientation we like, even upside-down.

to up-side-down
seth 180
house
end

Notice that all our forward and back commands, which make the turtle move, come in multiples of 10. We can use this idea, and modify our collection of procedures so that we can change the size of the house. We do this by introducing a variable, which I will call :unit. Here are the modified procedures. Because each one uses the :unit variable, it is modified in its procedure-name, as well as in each forward and back command it uses. (Making this kind of change is easier if you type the first change — turning the last 0 of a FD command into * :unit — and then select and copy this new material, and paste it on top of each of the last zeroes in the rest of the fd and bk commands. Incidentally, the * symbol is Logo's multiplication sign, so we are swapping the multiple of 10 for a multiple of :unit.)

to houseA :unit
to wall1A :unit
to wall2A :unit
to doormatA :unit
to doorA :unit
to window1A :unit
to window2A :unit
to roofA :unit
end
to wall1A :unit
pd repeat 2 [forward 4 * :unit rt 60 fd 10 * :unit rt 120]
end
to wall2A :unit
pd repeat 2 [forward 4 * :unit lt 60 fd 6 * :unit lt 120]
end
to doormatA :unit
pu rt 60 fd 2 * :unit
pd repeat 2 [fd 4 * :unit rt 60 fd 2 * :unit rt 120]
pu bk 2 * :unit lt 60
end
to doorA :unit
Try running `house.1 2` and `house.1 4`. The variable `:unit` works like a scale-factor — very clever! This makes it easy to fit a lot of small houses into a long street. We can even use another variable to specify how many houses we want.

```turtle
to streetA :unit :howMany
pu setpos [-230 -100]
repeat :howMany [pd houseA :unit pu rt 60 fd 15 * :unit lt 60]
end
```

Try running `cg streetA 2 15`

Finally, what do you think would happen if we made another change: swapping each right turn command for a left turn, and vice versa, swap each left for a right. Think about it, first. Then write down what you think will happen, and then try modifying the procedures, or, better still, a pasted copy of the procedures, renaming each one with a new procedure name, such as this example, where the dot-S stands for "swap".
Can you extend this house drawing idea? Two storeys? A veranda? Different windows? A chimney? What about using SETC commands to set the color of the turtle's pen?

If everyone in the class makes an individual house, they can all be combined to create a village of different houses. But don't just invent your own house design, on the outside, using MicroWorlds. Investigate different house designs, such as Victorian, Federation, Californian Bungalow, Spanish Mission, and other historical eras of house-design. And make your own inside plans.

Final Note 1: Your can try out MicroWorlds by downloading a demonstration version for either IBM or Macintosh, from the LCSI (Logo Computer Systems Inc.) homepage http://www.lcsi.ca/

This “demo” version won't let you save your programming or other work you create. But you can select and copy your programming, and then go to a wordprocessing document, and paste, and save your programming — as wordprocessing!

Final Note 2: If you want to try these examples without having to retype them, contact me by e-mail and I will send back an electronic copy of this article. You can then select, copy and paste from my e-mail directly into the Procedures Page of a MicroWorlds project. I can also send other examples for you to try.