# Lab 3. Modular Code : Procedures

## Learning Objectives

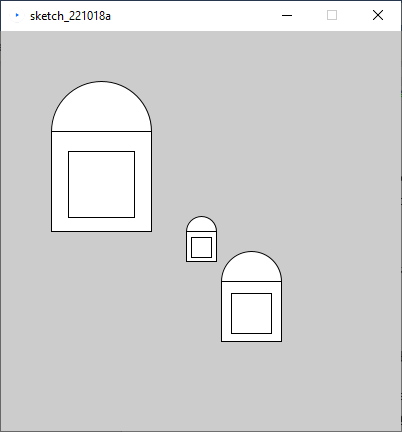
* Top Down Design
* Modularised code
  + Procedures
  + Parameter passing
* Setup
* Local variables

## Resources:

* Lecture Notes
* Processing.org website

Take a look at this example of a procedure and calls (they refer to the procedure as a void function) <https://processing.org/examples/functions.html>

**Pair Programming exercise**

In pairs write a program to produce the following pattern (similar- doesn’t need to be exact). It must make use of at least one procesure with parameters

**Ex1**. We are going to draw a simple motorcycle using procedures to draw the shape and the whole motorcycle.

1. As in the lecture, write a procedure (void) with **position** (x,y) and **size** parameters to draw a triangle as shown above, the height should be ½ of the size.
2. Write a procedure wheel with **position** and **diameter** parameters, to draw a black circle (for the tyre) and a smaller white circle on top for the hub.
3. Write a procedure to draw a motorbike, as shown above, with **position** and **size** parameters. Test it by drawing different size motorbikes in different positions on the screen.

Try the code below using different numbers (currently the text command is running slowly – looking for a file over the network).

void setup()

{

size(200,200);

numberBox(10);

}

void numberBox(int value)

{

rect(100,50,50,20);

fill(0);

text(value,110,65);

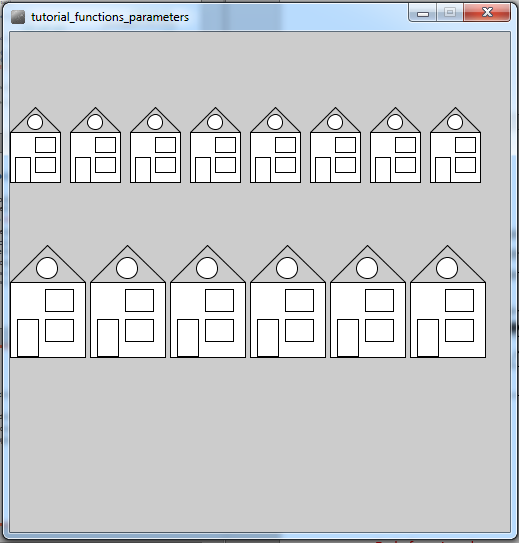
}

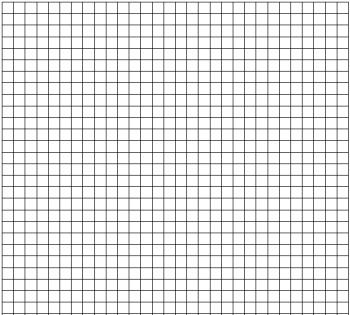
Ex 2. Amend the procedure above to draw this pattern in the centre of the screen, but of any size.

Ex 2B. Alter your procedure to be able to draw the pattern anywhere and of any size. Test it by drawing 3 of these in different places on the screen and each of a different size

Ex3. Extend your program to draw this pattern, again anywhere and of any size. You should add a new procedure, rather than modifying the previous version.

**Ex4.** (Pair Programming exercise) **Draw a street of houses**

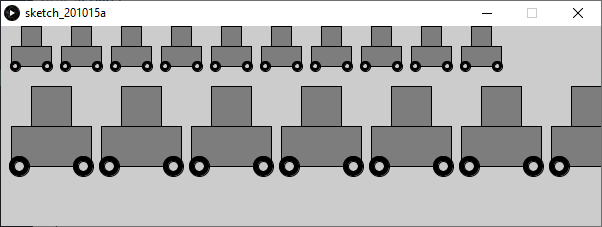
In the lecture we looked at the top down design to draw a terrace of simple houses. Take your design, and modify it to produce a street similar to the image (right) and implement the code. Should be 3 stages of top down design. We should be able to set the number of houses (in each terrace), the location and the size of the houses via parameters. Implement a square drawing procedure (you can make use of the **rect** command) and reuse any suitable procedures from the previous exercise.



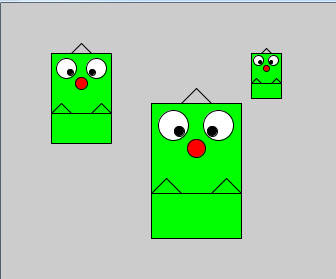
**Exercise 5. Traffic Jam**

How do we produce this motorway picture?

Use top down design with comments, then code it.



In the lecture we looked at the top down design to draw a terrace of simple houses. Take your design, and modify it to produce a street similar to the image (right) and implement the code. A good solution should include 3 stages of top down design, multiple procedures with parameters and be generalised so the image could be altered in future with minimal changes. We should be able to set the number of cars (in each row), the location, size etc .

**Extension Exercise**. Draw 3 monsters, similar to the image right, using a monster procedure (again with position and size parameters). Good modular code should be easy to read and understand which bit of code produces which part of the picture.

Don’t forget the **fill** (and **stroke**) command is like dipping your paint brush in a colour so everything after a fill is in the same colour, until you change it with another fill.

End of session please ensure that you have completed all the exercises for next week.

**Extension exercise** – read through and try out some of the examples in this tutorial on writing procedures and functions. In this source they just use the word “Function” to mean both, some sources call procedures “void functions” and some “methods” which we’ll come to later

<https://happycoding.io/tutorials/processing/creating-functions>