# MPHYG001: Coursework #2

Due on Monday, February 29, 2016

**Maria Ruxandra Robu- 14042500**

BoidsSimulator is a Python package that simulates the flying patterns of a flock of birds. By inputting different values for parameters (i.e. number of boids, collision alert, strength of flight to center), the package generates an animation that illustrates the simulated behaviour. The package can be used from the command line or as a library. More documentation on the installation can be found in the README file.

# Task 1

**Code smells, with reference to the commit log**

Refactoring smells:

- raw numbers appear in the code - I replaced the magic numbers with constants (commit 068564f)
- fragments of repeated code appear - I replaced the repeated code with functions (commit 3d18084)
- code needs a comment to explain what it does - I changed the names of the variables (commit 068564f, 16114d7)
- an expression becomes too long - I separated the lines into local variables and broke the expressions into separate lines of code that were less than 79 characters - using pep8 linter (commit 72728c1, 2dd138c)
- someone else had done this before - I replaced the hand-written code with library code (commit 16114d7)
- a global variable is assigned and then used inside a called function - I replaced the global variables with function arguments (commit cd98a4e, 84cbcb6, ddff7b4)
- two neighbouring loops have the same for statement - I merged neighbouring loops (commit cd98a4e, 84cbcb6, ddff7b4)
- a function of subroutine no longer fits on a page in the editor - I broke the whole function into a class with smaller units (commit cd98a4e, 84cbcb6, ddff7b4, 5ef73e9)
- code difficult to locate and follow - I separated the code into modules and classes (commit 3d18084, 5ef73e9, 3538efb)

Note: the logic of the main functions that simulate the behaviour of the flock (move towards the center, avoid collisions and match speed with formation) are based on the examples that we had in the slides.

# Task 2

**UML of the final class structure**

See figure 1.

# Task 3

**The advantages of a refactoring approach to improving code**

Refactoring is a way to make your code easier to understand and to follow. Strategies like breaking the code apart in small units that differ from each other based on functionality and creating variable names that

Task 3 continued on next page...

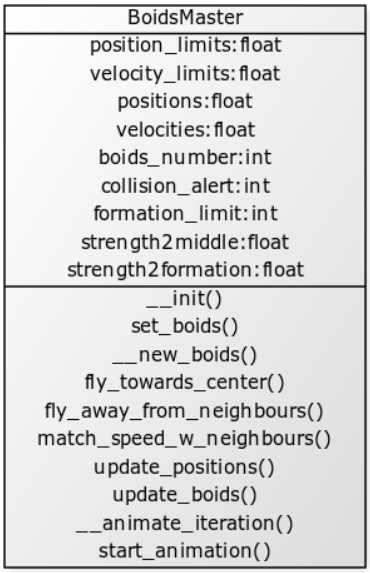


Figure 1: UML class design

explain the concepts, help other developers build on top of the project. Furthermore, a project that is well documented and refactored to be self-explanatory is much easier to debug. Refactoring can also speed up the coding process, since the developer does not need to spend time understanding the state of the project, because all the concepts are well embedded in the variable names.

# Task 4

**Discuss problems encountered during the project**

After each small change in the code, the regression test was run. After replacing the for loops of the functions for the behaviour of the flock (i.e. moving towards the center, flying in formation), the initial regression test failed with small differences (figure 2). This was ultimately solved by generating new fixtures for each case and using it as a regression test in the following refactoring process(figure 3).

```
[dhcp-153-98:refactoredBoids ucaHome$ nosetests
.F...
======================================================================
FAIL: Regression test - Tests that the class written for the boids works with th
e initial fixtures
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Applications/Canopy.app/appdata/canopy-1.5.5.3123.macosx-x86_64/Canopy.
app/Contents/lib/python2.7/site-packages/nose/case.py", line 197, in runTest
    self.test(*self.arg)
  File "/Users/ucaHome/Documents/Modules_MRes/PythonCode/Coursework2/bad-boids/r
efactoredBoids/tests/test_boids_refactored.py", line 32, in test_refactored_clas
s_boids_regression
    assert_almost_equal(after_value, before_value, delta=0.01)
AssertionError: 73.15072022064844 != 73.16123543741034 within 0.01 delta

----------------------------------------------------------------------
Ran 5 tests in 0.217s

FAILED (failures=1)
```

Figure 2: Failed regression test

```
[dhcp-153-98:bad-boids ucaHome$ nosetests -v
Regression test - Tests that the whole class works - fixture_general ... ok
Unit test - Fly to center method - fixture ... ok
Unit test - Fly away from neighbours method - fixture ... ok
Unit test - Match speed with neighbours - fixture ... ok
Test - the number of boids has to be an integer ... ok
Test - the number of boids has to be positive ... ok
Test - the collision alert has to be an integer ... ok
Test - the collision alert has to be positive ... ok
Test - the formation limit has to be an integer ... ok
Test - the formation limit has to be positive ... ok
Test - the strength to middle has to be a float ... ok
Test - the strength to middle has to be positive ... ok
Test - the strength to formation has to be a float ... ok
Test - the strength to formation has to be positive ... ok

----------------------------------------------------------------------
Ran 14 tests in 0.597s

OK
```

Figure 3: Final tests