Refactoring the Bad Boids

Tamara Kohler

Student number: 16049393
Course code: MPHYG001

1 Introduction

This report answers the questions required to complete the 'Refactoring the Bad Boids' assignment as part of MPHYG001. The git page for the project can be found at: https://github.com/TamaraKohler/boids.

2 Refactoring smells

1. *Smell: Raw numbers appearing in the code*
   Having raw numbers in the code rather than variable names hinders readability, and if a repeated value needs to be changed every instance must be updated which can lead to bugs. This issue can be fixed by replacing raw numbers with constants which have sensible variable names.
   Git commit: cba2e037cbd399766fdb245d7cd5ce2bcd4779e4

2. *Smell: Fragments of repeated code appear*
   This is an issue because if code needs to be updated then every instance of the code must be changed. This is time consuming, and can lead to errors. Replacing repeated code with functions can fix this issue. Well-named functions can also improve readability.
   Git commit: b8b473fc2120628b3c685f0db550e384cf796179

3. *Smell: A function needs to work corresponding indices of several arrays*
   Using numpy arrays instead simplifies the code, and increases functionality.
   Git commit: 125899593cabc64fc25bf03aa7004a1cbd3c9dfb

4. *Smell: Very large, complex functions*
   Functions which are very long and complex can be difficult to debug, and can decrease readability. This can be fixed by breaking long, complex functions into smaller, more simple ones.
   Git commit: 045d6140e3cbf089e8f8d4c4a487e92a59ee716a

5. *Smell: A global variable is assigned and then used inside a called function*
   This can be fixed by replacing global variables with function arguments.
   Git commit: 6c93b474dbda4913fa5e50c4a688f99ee4733428

6. *Smell: A lot of related functionality related to the boids*
   Related functionality should be kept together in a class to increase readability, and to make code easier to debug. Functions should be replaced with methods, and method arguments should be replaced with class members.
   Git commit: 268d4c1705cbbaed9e3368397b049fd710c4ac32

7. *You need to change your code file to explore different research scenarios*
   Storing constants in a configuration file is preferable to declaring constants within the code file as it gives the ability to explore different research scenarios without changing the code.
   Git commit: 395432ca755070dc69cdc2d3395208427ec00f31

8. *Smell: You find it hard to locate a piece of code*
   Keeping all code for a package located in one file makes it difficult to locate code in order to correct and debug. This issue can be fixed by breaking code into files / modules.
   Git commit: 42a36e2eefd76e6b4c0bc1887c83e1a43545bfc1

3 UML diagram

The YUML model for the Boids class is given in Figure 1.

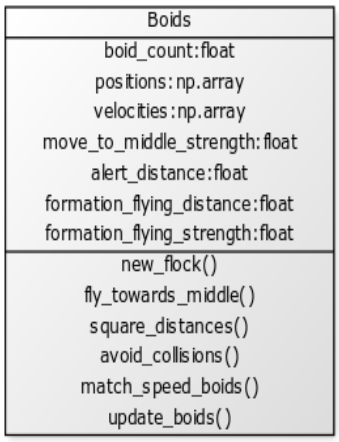

Figure 1: YUML model of the Boids class structure

An inheritance model is given in Figure 2.

Figure 2: Inheritance model for the Boids class

4 Advantages of refactoring

Refactoring is a technique for improving the internal structure of a piece of code, without affecting its external behaviour. The principle of refactoring is that code 'smells' are identified that alert you to the fact that the code isn't optimal, and each 'smell' is fixed by carrying out a refactoring, where each refactoring fixes an individual problem. Each refactoring is small, but carrying out a number of refactorings can significantly improve the structure of a body of code. There are a number of advantages to the refactoring approach:

- As each refactoring is small, it is less likely to introduce errors than if you simply tried to rewrite a large body of code, fixing a number of problems at once. Additionally, any errors which are introduced are likely to be easy to spot and correct, as the changes to the code will be small and localised. Carrying out refactoring along with good version control helps to ensure that refactoring is a safe way to improve code without changing its functionality.
- Refactoring code should improve code's readability. If code is well structured, with sensible names for variables and functions then the purpose of the code should be clear to anyone inspecting it.
- Improving the readability of code also makes code easier to maintain, as bugs are easier to fix when the purpose of a piece of code is clear. In addition, code needs to be well structured into functions / classes in order to unit test, and good testing is closely related to maintainability.
- Refactoring code also improves its extendability. If code is well broken down into functions and classes then code can be reused elsewhere, and the capabilities of the code can be extended more easily.

5 Problems encountered

There were two key problems that were encountered when carrying out the refactoring of the bad boids. The first of these was related to the regression test, which failed after the iterations in the initial code were replaced by numpy methods. This occurred because in the original code the match_speed_boids() function updated the velocities of the boids one by one, so the $(n+1)^{th}$ update depended on the n^{th} update, whereas in the numpy version all the updates are carried out simultaneously, and are hence independent. This means that although the code 'smell' was a valid issue, the refactoring did slightly change the functionality of the code. In this case I decided that the change was a subtle one that did not affect the code's overall performance, and that the improved readability of the numpy

method was more important than the small change in the boids' velocities, and I updated the regression test to encode the new result.

The second issue was related to the configuration file. The configuration file was added during the refactoring process to remove constants from the code, and while running the code in python the configuration file worked well, however when the code was pip installed and run from the command line there were some issues with finding the configuration file. This was resolved by adding a MANIFEST.in file to the package, and including the include_package_data = True command in the setup.py file, after which the code ran from the command line with the configuration file as expected.