INF-1400
Mandatory assignment 2 - Boids

February 23, 2017

1 Introduction

In this assignment you will create a simple simulator, more specifically a clone of the classic flock simulator boids, originally written by Craig Reynolds in 1986.[1]

The implementation will be written in **Python 3**, using the pygame library, with a goal of using the principles of object-oriented programming that we have gone through in class so far. That means you will implement the game using classes, methods and certain other requirements.

2 Implementation

2.1 Game rules (or: functional requirements

The simulator should simulate a moving flock consisting of *boids* operating by a set of rules. The goal is to make the flock move in a lifelike manner, without hard-coding the movement. Each boid has a set of adjacent boids, and should follow these rules:

1. Boids steer towards the average position of local flockmates.
2. Boids attempt to avoid crashing into other boids.
3. Boids steer towards the average heading of local flockmates.

The simulator should also include some additional features:

- Obstacles the boids need to avoid.
- Predators (hoiks) that will try to eat the boids.

The appearance of the simulator is up to you. The visual elements can be represented by sprites (images) or by simple shapes like circles or rectangles. The number of boids is also optional, however the size of the flock should allow it to split into several smaller flocks.

[1] https://www.red3d.com/cwr/papers/1987/SIGGRAPH87.pdf

2.2 Non-functional requirements

1. Make an object-oriented design (class diagram) in UML. Must be included in report.
2. Your solution must use at least one of the object-oriented techniques from class. For example inheritance, polymorphism, abstract base classes, etc. Explain and discuss your design in the report.
3. Your design shall include your own exception, for example *VectorException*, that handles exceptions that can occur in vector calculations.
4. If you use the precode, replace *raise Exception* with your own exception.
5. Hoiks and obstacles must be implemented.

2.3 Extra

Some suggestions for additional features:

1. Hoiks can eat boids and gain size.
2. Bait that attract boids.
3. Use the mouse to add boids, obstacles, bait or hoiks.

2.4 Hints

Make sure that the boids always keep a steady speed, preventing them from halting, or speeding off the screen. Write the code so that the significance of each rule can easily be modified. You should experiment with these, in order to maintain a life-like flock movement.

Suggested use of object oriented techniques:

- Both boids and hoiks could inherit from a moving_object class.
- All visible objects could inherit from a drawable_object class.

We suggest that you use the vector class from the precode.
Check out these web-sites for more information:

- http://www.kfish.org/boids/pseudocode.html
- http://www.red3d.com/cwr/boids/
- http://en.wikipedia.org/wiki/Boids

2.5 Precode

To help you get started, you have received some precode. The precode includes a vector class to handle position and speed, as well as methods for handling collision between objects (circles and rectangles).

It is not required that you use this code, feel free to roll your own. Note: If you use the precode, replace *raise Exception* with your own exception.

3 Report

The report should describe your implementation. Show that you understand how your implementation works. Remember that the code should be well enough commented and written so that it, in addition to the report, makes it easy for another person to understand how your program is put together. The report *must* be delivered in PDF format.

4 Hand-in

A student with student-id *qwe123* puts the files in the following directory structure:

```
inf1400-qwe123-2/
        |--src/
        |   |--all the source files here
        |   |--README
        |
report.pdf
```

The directory `inf1400-qwe123-2` is compressed to a zip archive and is handed in through Canvas before the deadline. The README should contain information on how to run the program.

Note that the report pdf should be **outside** the zip file. This is to simplify our plagiarism check.

Deadline: 20.03.18 at 23:59

5 Cheating

Remember that cheating, or attempted cheating during mandatory assignments, is considered the same as cheating during an exam. Here are some guidelines.

- Getting *help* from another student to solve a problem is allowed.
- Discussing design and code problems with other students is allowed.
- Copying code is **not** allowed.
- Copying the design off someone or off the internet is **not** allowed.
- Wrong use of/missing references are **not** allowed.
- Getting the *solution* (code, design, or report elements) is **not** allowed.