INF-1400
Mandatory assignment 2 - Boids

February 19, 2021

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. The implementation will be written in Python, using the pygame library, with a goal of using the principles of object-oriented programming as best possible. That means you will implement the game using classes, methods and inheritance.

2 Implementation

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.

2.1 Requirements

1. Implement the simulator in accordance with object-oriented design, using objects and classes.
2. Inheritance must be used to implement at least one class.
3. The simulator must follow the rules described above.
4. Hoiks and obstacles must be implemented.
5. The report must give a description of inheritance in object-oriented programming, and how you have chosen to use this feature.
6. The hand-in must include a class diagram (as shown in lectures) which describes relations between the different classes.

2.2 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.
4. 3D

2.3 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 inheritance:

- 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 Pygames Vector2 module.
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

2.4 Precode

For this assignment, we have chosen to exclude the precode. The reason for this is that intersection between objects isn't a vital part of the assignment. If you find it necessary, you may use the precode from the previous assignment to check for collision.
Feel free to use any modules/libraries that doesn't trivialize the tasks of the assignment. Remember to include how to install these (if that is necessary) in the README.

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
report.pdf
```

The directory `inf1400-qwe123-2` is compressed to a zip archive, containing a copy of the report, and is handed in via Canvas together with the report before the deadline. The README should contain information on how to run the program.

5 Peer-Review

After the deadline there will be a review where you and another student will look at each others code and solution in order to come up with constructive feedback on e.g. structure, flow, implementation, etc. At the next colloquium session you will exchange feedback, with the TA present as a moderator.

What to look for in a code review:
`https://google.github.io/eng-practices/review/reviewer/looking-for.html`

Deadline: 11.03.21 Before 23:59

6 Cheating

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

- 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 *help* from another student to solve a problem is allowed.
- Discussing design and code problems with other students is allowed.
- Getting the *solution* (code, design, or report elements) is not allowed.