INF-1400: Object-orientated Programming
Mandatory assignment 2 - Boids

Eskil Bjørnbakk Heines

February 28th, 2023

1 Introduction

In this assignment, the task is to create a simple simulator being a clone of the classic flock simulator Boids originally written by Craig Reynolds in 1986. There is no pre-code given with the assignment, making it very open in how to structure and implement the code. The main goal is obviously to make a simulator cloning the original Boids simulator, but while still using the principles of object-oriented programming, like classes, methods and inheritance, and coding it in the pygame library. The main requirements for this assignment are:

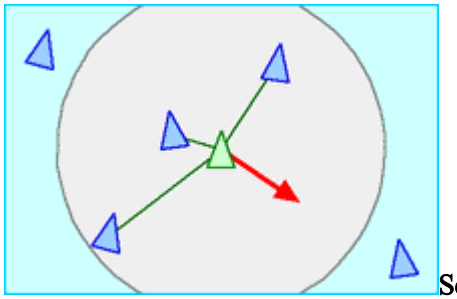
- Implement the simulator in accordance with object-oriented design, using objects and classes
- Inheritance must be used to implement at least one class
- The simulator must follow the rules of the boid simulator
- Hoiks and obstacles must be implemented

2 Technical Background

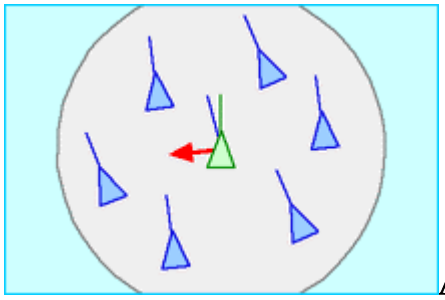
In this assignment, it is essential to use the pygame library to code the boids simulator. Pygame [1] is designed to specifically write video games with different Python modules. It includes computer graphics and sound libraries designed for Python programming. It is a replacement for the earlier PySDL [2]. Along with pygame, comes a great feature called Sprites. Sprites [3] allows the programmer to make classes with visible game objects that can be moved around on the screen with animations. And for making animations, vectors and vectors mathematics must be implemented.

One of the requirements of the code is to use inheritance in the classes. Inheritance allows the class to inherit all the variables and methods from another class. The class that inherits the variables and methods is called the child class, while the class it inherits from is the parent class. Since the boids and hoiks in this code will behave in a similar fashion, making inheritance a great tool to make this easier to code.

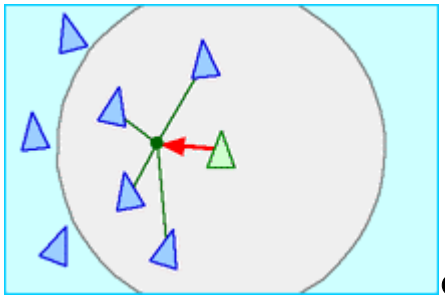
The algorithm for the boids simulation has three main rules which need to be followed for it to work; separation, alignment and cohesion. This is what these rules is supposed to do (taken from one of the hints [4]):



Separation: steer to avoid crowding local flockmates



Alignment: steer towards the average heading of local flockmates



Cohesion: steer to move toward the average position of local flockmates

With these rules applied the algorithm should look something like this [5].

3 Design and Implementation

In this assignment the task is to implement the boids flocking algorithm with hoiks and obstacles included. The hoiks are supposed to hunt the boids, and the boids should escape the hoiks and avoid the obstacles. The hoiks, boids and obstacles are each an individual class, with the boid and hoik having inheritance from a different class, vehicle. The vehicle class is there because the boids and hoiks have a lot of similarity in variables and how they move. Both should have an update method that's updating the acceleration, velocity and position, as well as a method that makes them reappear on the screen if they go outside the screen.

3.1 Vehicle

The vehicle class is the parent class of the boid and hoik class. In the vehicle class all the necessary variables are defined, as well as the method for animating the variables and making the objects not disappear when they go outside of the screen. This class makes it easier to implement the boid and hoik class without having to code methods twice.

3.2 Boids

In technical background, the principles behind the boids flocking algorithm were explained. The cohesion method uses the principle of a given perception of the boid, like what the boid can sense around itself. There is also a vector used to give the boid acceleration towards the other boids in the local area. The alignment method does the same in having a perception of that the boid can sense but has a vector that makes the boid head in the same direction as the local boids in its area. The cohesion method has a vector that makes the boid steer towards the average position of all the other boids in its local area. For avoiding the hoiks and obstacles a method that like the separation method is implemented. Finally, the flock method that's combining all the methods into one method making the code easier to understand.

3.3 Hoiks

The hoiks behave in a similar manner to the boids, only that they do not have the same rules applied to them. They do not align, cohere or separate from each other, but rather hunt the boids. This method is implemented with the same structure as the separation of the boids, meaning it's a vector that accelerates the hoik towards the local boids, like it's hunting them.

3.4 Obstacles

The obstacles are small circles randomly spawned to the screen with the purpose of being an obstacle for the boids. Obstacles were a requirement for the assignment and was one of the more complex algorithms to implement. It is like the separation of the boids, but it took a long time before the solution was clear. With inspiration from the internet, specifically this page [6], the necessary function for making the method work.

4 Evaluation (optional)

This code is far from an optimal and perfect boids simulation. A lot could have been done better and easier. This is of course things that takes time to perfect, and with the time given to finish the assignment and the requirements, it's not expected to be perfect. The separation of the boids is the one I think should have been done better since I don't feel it works very well.

5 Discussion (optional)

The code will sometimes crash with the error being "ValueError: Cannot scale a vector with zero length", something I have not figured out why it does. Sometimes the code can run for a long time without problems, and sometimes the code crashes after only a few seconds. This is because of the "scale_to_length" function that's used is some of the methods. Have not found a better way to solve this problem.

The separation of the boids is not perfect, since sometimes they overlay each other instead of separating. This is not a big problem but needs to be addressed.

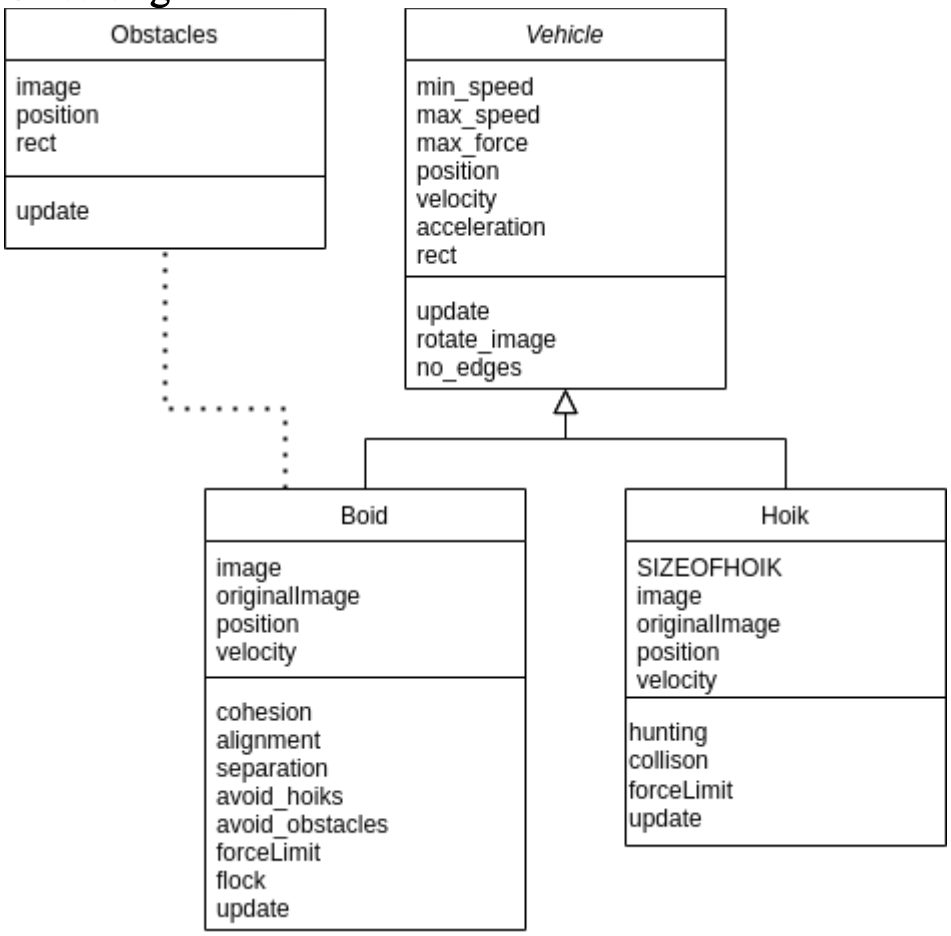
6 Conclusion

The tasks and requirements in the assignment are implemented and done. The code is running as it should be except for the value error and separation.

Acknowledgement (optional)

Worked with my present classmate Kristian Harvey Olsen (1st year of Informatikk Bachelor) for this assignment
Have used a YouTube-video [7] as inspiration to implement the algorithm, making some of the code similar, but still a lot of differences because of the fact it's two different programming languages.

Class diagram



References

[1] Pygame, Pygame library:
https://www.pygame.org/news

[2] PySDL, earlier Pygame:
https://pysdl2.readthedocs.io/en/0.9.13/

[3] Sprites, Pygame sprites:
https://www.pygame.org/docs/ref/sprite.html

[4] Hints from assignment, pictures and definitions of rules, red3d.com:
https://www.red3d.com/cwr/boids/

[5] Boids visualization, youtube.com:
https://www.youtube.com/watch?v=tuXBikW3GyE

[6] Boids implementation as inspiration, github.com:
https://github.com/Doonse/BoidsSimulation

[7] Flocking simulation, youtube.com:
https://www.youtube.com/watch?v=mhjuuHl6qHM&t=270s

1]