Scope of the project

The project will have three fairly distinct parts to create the simulation:
Rendering the world:

Ideally the engine can support rendering upwards of thousands (or more) of individual boids, obstacles, and toggleable velocity/heading vectors and “thoughts” of boids. The world will also optimally include dynamic placing and removing of new boids and new obstacles.
The camera should also support zooming, free movement, panning and follow-functionality, to follow individual boids
Algorithms:

The initial plan is to implement all algorithms in Reynolds’ paper, some of which are: Seek, Pursuit, Arrival, Obstacle avoidance, Wander and Flocking

The plan is to also have multiple types of “brains” that implement (or weigh) the implemented behaviors differently, for example to create “predator” boids and “prey” boids, which prioritize seeking and fleeing, respectively.
Possibly also implement some kind of a “eating” interaction, where a boid touching another boid would remove the boid that was hit, essentially allowing for simulating “cat and mouse” -type of behavior.
An end goal is to implement gpu assigned calculations to help with large numbers of boids.

Boids and other objects:

The boids are currently formulated to fit under a virtual master class, which will host both the boid vehicles themselves, as well as other objects in the constructed world. With regards to the boids; the current plan entails creating a vehicle class for the boids, which will gain any global values such as accelerations and turn speeds for boids. It will not, however, be in charge of any of the selections with regards to movement. The boid vehicle will extend a brain class, which will determine the behaviour of the boid. A brain class contains all the basic steering functions of a boid such as seeking and wandering, while the inheriting subclasses will contain flavours such as a flocking boid, or fleeing.

The current structure is still fairly abstract, and as such is at a position to change - potentially even radically so.

Major architectural decisions

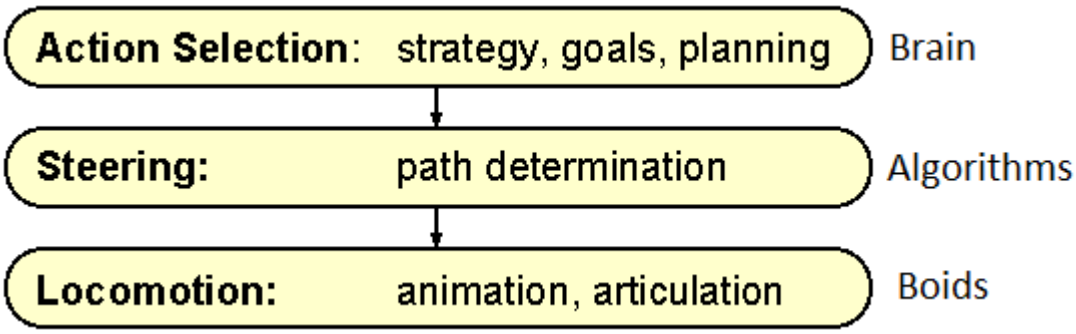


Fig 1. Reynolds’ hierarchy and our projects’ planned implementation

The whole group agreed on creating the project in three dimensions, as the two-dimensional version seemed fairly straightforward to implement.

The implementation of individual boids will be kept as simple as possible to allow for faster computations and thus allowing larger amounts of boids to be simulated at once. Depending on observed performance down the line other optimizations might have to be done on algorithm and rendering layers.

We also opted to implement our own vector and matrix classes and their operations, mostly to learn optimization and implementing mathematical models in code. Compared to a off-the-shelf library that implements matrices, our implementation will most likely be stripped-down version of the concept, as we will only include the operations we seem necessary.

For rendering the project we opted to go for a readymade 3D capable engine (specifics undecided as of writing), as making our own 3D rendering engine that would support our needs would likely take most of our time, and isn't the core part of the project.

Implementing GPU-accelerated computing is also something we are interested in, but depending on the perceived difficulty of implementing it, we might attempt to implement it (most likely by using OpenCL), although considering the time limit, it's also probably we will settle on using the CPU for our calculations, which puts a higher emphasis on optimization in other areas of the code.

Schedule

We'll try to avoid the classic last-minute crunch, and instead work on the project steadily every week. Priorities for the project in the beginning are as follows:

1. Getting the initial testing environment running
2. Getting the vector, matrices and basic boids working
3. Implementing the algorithms
4. 3d rendering
5. Nice-to-haves, like optimization, "eating", dynamic object adding etc.

Optimally we should have the basic testing environment (and something in it) in the first two weeks (45-46), and some of the basic algorithms working in the following week (47). After that most of the progress will be incremental, with adding most of the algorithms and eventually getting the 3d renderer working, hopefully no later than start of week 49.. Most of the boid logic and algorithms should be done by then, which leaves around 10 days to finish up all the miscellaneous stuff.

Distribution of roles

The current distribution of roles has been done according to personal interest for the project members. Currently the plan is for everyone to begin working together on getting a testing

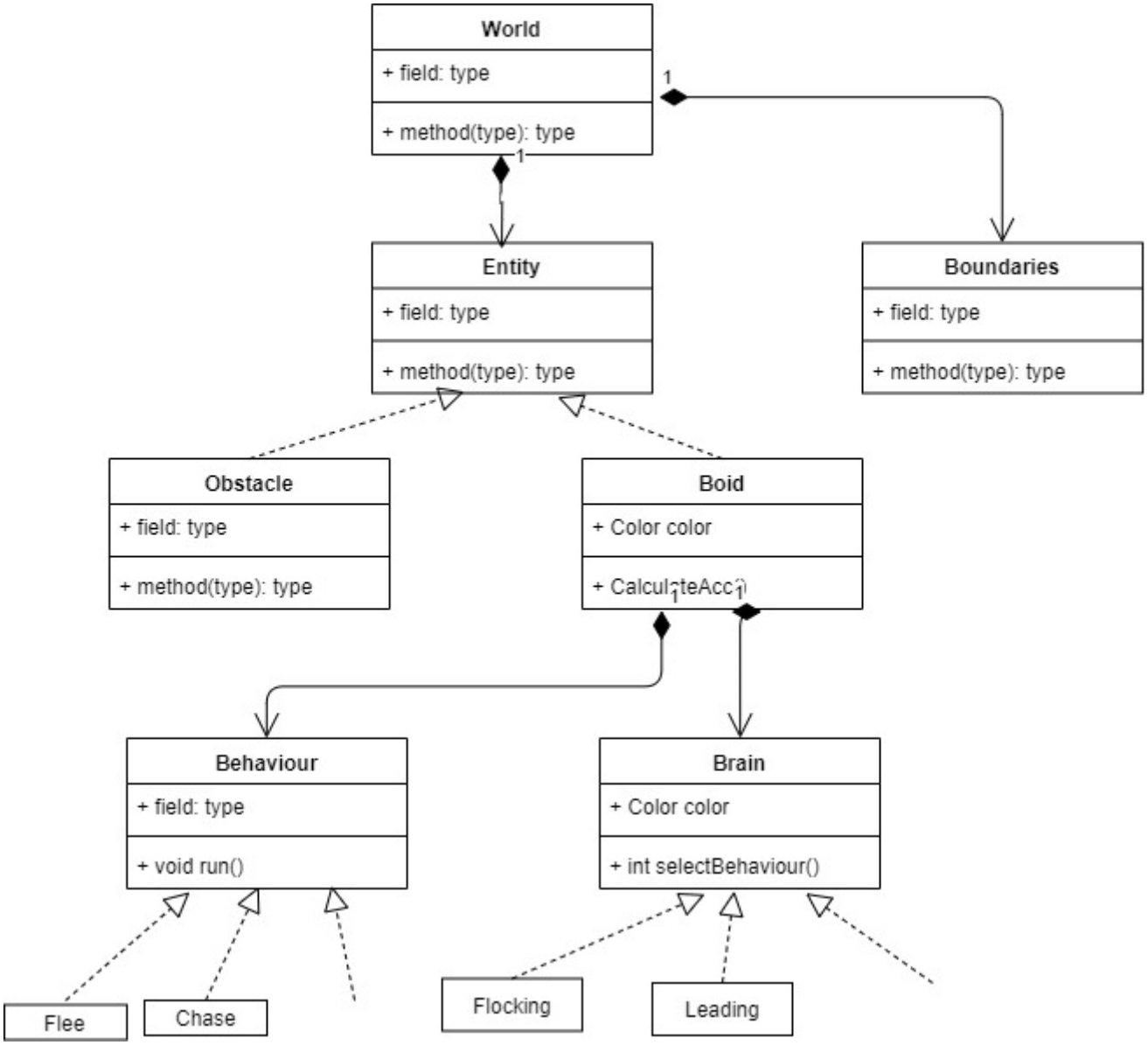
platform ready, after which it will be possible for the group to separate to their own assigned sections.
The inital assigned roles we have for our group are the following:

SFML basic testing environment:	Sakari
Boid algorithms:	Emma
Vector basis for calculations:	Sampo
Final 3d testing environment:	Joonas

From the previous roles Sakari will move on to assist with the boid algorithms once the initial testing environment is completed, however, this may still be altered depending on the scale of the workload for the 3d testing environment. Similarly Sampo will move on to implementing the boids in full once the vector basis for the code is complete.

Design rationale

The general ideas behind our plans so far come from the documentation on boids provided with the assignment. At its core the project plan is fairly ambitious and we are looking to begin work as soon as possible. As such it will not be an incredible loss, should we fail to complete implementation of some higher level functions for the boids, as if we manage to create a 3d-basis and achieve gpu utilization for calculations..


World
+ field: type
+ method(type): type
1
1
Entity
+ field: type
+ method(type): type
Boundaries
+ field: type
+ method(type): type
Obstacle
+ field: type
+ method(type): type
Boid
+ Color color
+ CalculateAcc()
1
1
Behaviour
+ field: type
+ void run()
Brain
+ Color color
+ int selectBehaviour()
Flee
Chase
Flocking
Leading