Universiteit Leiden

Master Computer Science

FishEcoModeler: A performant, simple to use and extensible open source framework for simulating fish populations in complex 3D environments

Name:	Niels Witte
Student ID:	2685620
Date:	01/08/2023
Specialisation:	Artificial Intelligence
1st supervisor:	Dr. M. Preuss
2nd supervisor:	Dr. H. Wang

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Over the years there have been multiple attempts at modelling marine ecosystems. Use cases include ecological studies, fish farms, large-scale fishing operations or even video games. Complex mathematical models such as the Wisconsin Fish Model [23] and systems such as Ecopath [10] have been created to simulate these ecosystems. Although these systems are biologically accurate, they often face problems with accessibility, scalability or extensibility [31].

This thesis tests the statement that bio-energetic models are too complex for simulation purposes[6] and describes the design, implementation and use of a novel 3D simulation framework for fish populations named *FishEcoModeler* or *FEM* which is designed to be performant, user-friendly and extensible by being available for modification and inspection as an open-source project on GitHub.

It is shown that *FEM* is capable of simulating large schools (1000) of fish in complex 3D environments. Experiments show that it can accurately and rapidly simulate existing bioenergetics models and the effects of environmental properties such as temperature on the fish population.

Acknowledgements

First I would like to thank my supervisor, Dr. M. Preuss for his guidance and support throughout the project. His experience with guiding other students through their projects has been invaluable and I am thankful for his patience. I would also like to thank him for being the instructor of the *Modern Game AI Algorithms* course that inspired this project.

Next, I would like to thank Dr. H. Wang, my second supervisor, for introducing me to swarm-based algorithms in his course *Seminar Swarm-based Computation with Applications in Bioinformatics*, which directly inspired the implementation of the swarm-based fish behaviour in the simulator.

Finally, I would like to thank the authors of Fish Bioenergetics 4 [15] for their work on open-sourcing their software and implementing existing mathematical models in code, which has been a great help in the implementation of said models into the simulator.

Niels Witte, Leiden, August 2023

Contents

1 Introduction

This paper describes the design and implementation of *FishEcoModeler*, a 3D simulation framework for fish populations that is designed to be user-friendly, extensible and open-source[1]. The framework is intended for use by both researchers and students and has been designed with ease of use and understanding in mind. Its extensibility allows for the incorporation of new features and functionality, while its open-source nature enables broad use and collaboration.

FishEcoModeler is strongly motivated by a project created during one of the courses at Leiden University, during which an underwater environment was implemented using the Unity 3D game engine. This project directly led to a personal interest in ecological simulation and with the current state of the art being Fish Bioenergetics 4 [15], the framework was born.

The main goal of this framework is to capture complex environmental conditions that the fish are subject to, in addition to the incorporation of existing bioenergetics models. These models are often parameterized with environmental properties such as temperature, which is known to change based on depth and time of year. It is designed to be able to capture these properties and to be able to simulate the effects of these properties on the fish population.

Rather than a fully-fledged simulation tool, *FishEcoModeler* serves as a proof-of-concept. It exists to fill a gap in the current state of the art and to provide a starting point for future research. With this further research being done in cooperation with ecologists, it can be extended to become a powerful and complete simulation tool.

1.1 Research Questions

This work aims to explore and evaluate an alternative approach to the modelling of fish populations and to create a framework that allows researchers to further understand and predict the implications of environmental changes on fish populations. This thesis aims to answer the following research questions:

1. With modern technology, is it possible to create a simulation framework that is capable of simulating large schools of fish in complex 3D environments?
2. Can we preserve the accuracy and functionality of existing bioenergetics models whilst incorporating the effects of environmental changes?
3. How does the simulation framework compare to existing fish population simulation tools in terms of performance, ease of use, and ecological accuracy, and what unique advantages does it offer to researchers and educators in the field of ecology?

[1]https://github.com/Trottero/master-thesis-simulator

The next chapters of the thesis explore the implementation, design and use cases of *FishEcoModeler*. First in section 2 the current and past literature is reviewed. In section 3 the problem is formalized and in section 4 some basic ecological and bioenergetics concepts are explained with their implementation process being described in section 5, but the focus remains on providing a stable base for future research. The experiments in section 6 and their discussion are tightly coupled and therefore grouped. As this is mostly exploratory work, experiments are driven by previous results rather than thought out at the start of the thesis. The thesis concludes with suggested future work in section 7 and a reflection on the work done in section 8 together with a summary of the results and the research questions will be answered.

2 Related Work

This simulator combines problems and solutions from multiple fields of research. This section will outline the most important related work and a brief history and the current state of the art of fish simulation and bioenergetics.

2.1 Biomass Simulation

Simulation of ecosystems, and more specifically biomass has been a topic of research for a long time. In 1974 Kitchell, Koonce, Magnuson, *et al.* proposed a biomass simulation algorithm which would predict the total amount of biomass in an ecosystem [28]. Algorithms like these were later extended to account for more complex multispecies ecosystems such as marine ecosystems [31]. As Hanson, Johnson, Schindler, *et al.* outlined in their *FB3* paper[19], these ecosystem-wide simulators are limited in the way they compute the energy consumption of individual agents within the system, as they assume a population to be a single entity when in reality a population is made up out of different individuals, each with different energy characteristics.

2.2 Fish Bioenergetics

Due to their applications for fish farming and ecological studies, fish bioenergetics models are a commonly researched topic [6], [9]. In the past many of these models were too complex to be used for simulation purposes [6] but with powerful computers becoming more accessible than ever before, this is now possible.

Computer Models Johnson and Hewett in 1987 were already working on creating a general model for fish bioenergetics, in this work they propose a simplified version of existing models fit for simulation. This software, named *Fish Bioenergetics* or *FB* targeted the Apple IIe and IBM computers, distributed using floppy disks. This was later replaced with *FB2* [21] targeting newer IBM computers, boosting performance and adding newer models.

As computers evolved, so did the software. Hanson, Johnson, Schindler, *et al.* created *FB3* in 1997, a replacement for *FB2*. This software was written in C++ and targeted Windows 95/NT computers. It was designed to be more user-friendly and to be able to run on more modern computers. It also added support for more models and more parameters.

The latest instalment, and the current state of the art, is *FB4* [15]. The main drive behind this version was to modernize the software and make it more accessible to researchers. As *FB3* was distributed using disks, *paid* and incompatible with newer Windows 10 / 11 systems, it was not easily accessible to researchers. *FB4* is distributed as open-source software, written in *R*, and is available on GitHub[2]. In addition to modernisation, *FB4* also

[2]https://github.com/jim-breck/FB4

added support for newer models and new parameter presets which made it easier to configure the program.

In 2004 Neill, Brandes, Burke, *et al.* created an alternative to *FB3*, this model, *Ecophys.Fish* was created in the visual modelling language *Stella* [8]. Stella has been used for modelling economic and ecological systems [11]–[13] and has been a subject for comparison against other visual modelling languages [7]. Stella is however a paid software package, which limits its accessibility.

Rainbow Trout Bioenergetics The aforementioned software, *FB4*, supports a wide variety of 75 unique fish species such as Burbot (Lota lota) [36], Yellow Perch (Perca flavescens) [29] and Rainbow Trout (Oncorhynchus mykiss). The latter of which will be the primary focus of this thesis, as it is one of the most commonly studied fish species with a relatively well-understood bioenergetics model [39], [41], [50].

2.3 Fish Movement

The movement of fish has been studied quite extensively, with the primary applications being in engineering, rather than ecology [4] within these papers they attempt to model the skin of the fish and the way it interacts with the water. This could for example be used to create more efficient propellers for ships or reduce the drag of their hull [2].

Schooling Fish are known to present complex behaviour when in groups, this behaviour is known as schooling [37]. This behaviour is effectively a way for the fish to communicate factors in the environment to each other. The school collectively forages and avoids predators [1], [22].

Particle Swarm Optimization The schooling behaviour of fish also influenced other technologies such as Particle Swarm Optimization [16], where the foraging behaviour of the fish or other species which exhibit this grouping behaviour, is imitated to find an optimal solution in complex search spaces [24], [38].

2.4 Simulations

Simulations are a cost and time-effective method of doing experiments, provided that the simulator can accurately enough model the real world [25]. With computers doubling their transistors every two years for the past 50 years [33], [43] computers have become increasingly more powerful. A significant portion of the research on bioenergetics models was done at the start of the 21st century [9], when complex modelling was not feasible [6]. As is evident by the emergence of *FB4* and *Ecophys.Fish* with both of them having sub-minute compute times for a month, this is no longer the case.

Ecosystems Several methods of simulation ecosystems exist, but in the literature, ecosystems are expressed as mathematical models. Kremer and Nixon[30] show multiple examples of such models, and propose a new model which combines existing models for coastal ecosystems. This approach models the ecosystem as a whole, similar to the method proposed by Kitchell, Koonce, Magnuson, *et al.*[28].

Another approach to ecosystem simulation is agent or individual-based ecosystems [44]. Over the years many models have been proposed [14], [18], [40]. These models treat every individual within the ecosystem as an entity with unique biological properties, and allow for variety to exist within the populations of the ecosystems. This approach is more computationally expensive than the previous approach, as every individual needs to be simulated separately.

The predator-prey model is a common concept within ecosystem simulation [32]. In this model, the relationship between different species is expressed as follows; one species, such as chickens, are the prey, they feed on the resources available in the environment such as plants. Another species, e.g. the fox, is the designated predator, these hunt the prey species. This relationship creates an interesting effect in the population curves for both; when the prey population increases, the predator population increases as well, up until a tipping point where the prey has exhausted the environment, thus becoming unable to reproduce further, this then directly affects the predator population, which will decrease as well. This creates a cycle of population growth and decline for both species [3], which can be described as population-, predator-prey- or limit-cycles [3], [34], [49].

Simulation in Existing Software Several simulators already exist for ecosystem simulation, in 2021 Kiss and Pusztai created a visual ecosystem simulator, with the intent of generating more interest into ecosystem simulation. This simulator, named *Animal Farm*, received another revision in 2022[27]. This simulator allows for custom landscapes, and the simulation of multiple species, but is focused on terrestrial species.

Ecotwin as proposed by Glimmerfors and Skoglund in 2021 is another example of existing ecosystem simulation software, this software is focused on the evaluation of behavioural Artificial Intelligence models. It was created in *Unity* and is available on GitLab[3] and their website[4]. Like the previously mentioned simulator, this simulator is also focused on terrestrial species.

Lastly *EcoPath* is a simulator for marine ecosystems, this simulator is a *population-based model*, which means that it simulates the ecosystem as a whole, rather than simulating every individual within the ecosystem [10]. It can simulate large fish populations accurately over a long period.

It is worth noting that both *Ecotwin* [17] and that *Animal Farm*[26] did not exist at the

[3]https://gitlab.com/ecotwin/
[4]https://www.ecotwin.se/

inception of this thesis, and as such much of the work done in this thesis is independent of these simulators.

3 Problem

While extensive research has been conducted on Fish Bioenergetics and Fish Simulation, no existing software or framework combines these two elements.

To address this gap, this work proposes a framework that enables other researchers to test current and future fish bioenergetics models in a more intricate environment. While the primary focus is not on increasing the complexity of the environment, the framework can be readily modified to achieve this goal. Since the framework represents the integration of two distinct areas, definitions for both will be provided.

3.1 Fish Energy Consumption

Estimating fish energy consumption is a complex and challenging task, as it is influenced by a wide range of factors, including species, size, age, temperature, food availability, and habitat characteristics. Moreover, energy consumption is not a static quantity but can vary significantly over time and changes as its environment or ecosystem changes. This variability makes it difficult to accurately estimate fish energy consumption in the wild.

Estimating the energy consumption E of a fish over time t can be formulated as a function of the fish's weight W and the time-varying factors that affect its energy consumption, such as temperature T, food availability F, and other environmental properties and as such can be defined as in Equation 1.

$$E = f(W, T, F, ..., t) \tag{1}$$

3.2 Simulating Fish Behaviour

This aspect of the problem pertains to the behaviour that fish exhibit in a given environment. Such behaviour can be described as a set of rules that govern fish movement. These rules may be as simple as a fish swimming toward the nearest food source, or as complex as a fish following a leader while avoiding predators. In this thesis, the focus is on simulating the swarming behaviour of fish. The crucial aspect of this behaviour is that the fish work cooperatively to achieve the shared goal of foraging for food, which is the only means by which they can gain energy. This problem is defined in a similar way to prediction models, where given the current positions of the fish in the swarm, the goal is to predict the next position of each fish in the swarm. This can be formulated as in Equation 2.

$$P_{i,t+1} = f(P_t, t) \tag{2}$$

4 Background

As can be concluded from related work in section 2. The relevant fields of research for this thesis are quite varied and complex topics. This section will dive deeper into the specific methods that are used in the simulator, as well as the relevant background information for these methods. It is important to note that some biological concepts are simplified for the sake of brevity and clarity.

4.1 Modelling Energy Consumption

The general model for fish bioenergetics, also known as the Wisconsin fish model, was first proposed in 1987 by Johnson and Hewett. This model was created to estimate the energy consumption of fish in the wild, and it takes into account various factors such as water temperature, fish weight and size, food availability, and metabolic rate.

The Wisconsin fish model is based on the principle that the energy intake of a fish is proportional to its metabolic rate and the available food in its environment. This principle is used to calculate the amount of energy a fish consumes in a given period. The model also considers the loss of energy due to respiration, excretion, and other factors.

Another key factor of this model is that it is based on an individual, rather than an entire population. This allows for more accurate predictions of populations with a large variance in size and weight [19] and is generalized as follows:

$$E_{consumed} = metabolism + wastes + growth \tag{3}$$

Rainbow Trout is a species of freshwater fish that is native to the Pacific Northwest region of North America. Rainbow trout are widely used in aquaculture and have been introduced to other parts of the world, where they are popular among anglers. They are known for their energetic behaviour and their ability to thrive in a range of water temperatures and conditions. Rainbow trout are also a well-studied species regarding energy consumption and have been extensively researched in the field of fish bioenergetics.

It is not realistic to reimplement all of the functions present in *FB4* in the time frame of the thesis, and therefore the non-anadromous (non-migrating) version of rainbow trout (Oncorhynchus mykiss) was picked as an example fish, as it has a well-established and accurate energy consumption model that is available in *FB4* [15]. The following equations and later experiments all use the rainbow trout model as a basis for the energy consumption of the fish.

Extensions on the Wisconsin Fish Model Realistically, the Winsconsin fish model is an observed common factor with bio-energetic models for a plethora of fish species, with different fish species having to substitute different functions for the three components of the model. *FB4* [15] further splits this model into four components: *consumption*, *respiration*, *egestion*, *excretion* and a specific *Activity* modifier (Equation 4). These components are then used to calculate the energy consumption of the fish.

$$E_{consumed} = consumption_{max} - (respiration + egestion + excretion + activity) \tag{4}$$

Equation 4 is based on the max a fish could consume, minus the energy lost by waste. The max consumption is calculated by Equation 5. This equation is essentially the amount of food that's available relative to the weight of the current fish. This statistic is used in combination with the predicted consumption value (Equation 7) to create what is known as the p-value (Equation 6). This value is treated as the proportion of the maximum consumption that the fish is expected to consume and is used in the waste components of the rainbow trout model.

$$consumption_{max} = g_{pray}/g_{fish} \cdot density_{pray} \tag{5}$$

$$p = consumption_{max}/consumption \tag{6}$$

The *consumption* function in Equation 7 which is used in *FB4*[15] was originally proposed by Thornton and Lessem in 1978. This consumption function is parameterized by the water temperature (T), the coefficient between water temperature and consumption (CQ), the optimal observed laboratory temperature (CTO), the maximum temperature at which consumption ceases (CTM), the temperature at which consumption is a custom reduced fraction (CTL and $CK1$) and a lower bound fraction ($CK4$). Finally, the allometric mass function (C_{max} *parameterized by* CA *and* CB) is used to calculate the maximum consumption. This function describes the relationship between the current weight of the fish (W) and the maximum amount that it could consume based on said weight.

$$\begin{aligned}
CG1 &= \frac{1}{CTO - CQ} \cdot \log \frac{0.98 \cdot \langle 1 - CK1 \rangle}{CK1 \cdot 0.02} \\
CG2 &= \frac{1}{CTL - CTM} \cdot \log \frac{0.98 \cdot \langle 1 - CK4 \rangle}{CK4 \cdot 0.02} \\
L1 &= e^{CG1 \cdot (T - CQ)} \\
KA &= \frac{CK1 \cdot L1}{1 + CK1 \cdot (L1 - 1)} \\
L2 &= e^{CG2 \cdot (CTL - T)} \\
KB &= \frac{CK4 \cdot L2}{1 + CK4 \cdot (L2 - 1)} \\
C_{max} &= CA \cdot W^{CB} \\
consumption &= C_{max} \cdot KA \cdot KB
\end{aligned} \tag{7}$$

The *respiration* function as implemented in *FB4* is displayed here as Equation 8. This equation was first introduced by Kitchell, Stewart, and Weininger in 1977 and is parameterized by the temperature (T) and the weight of the fish (W). The model then uses the following parameters to compute the loss due to respiration; the optimum temperature for respiration (RTO), the maximum lethal temperature (RTM), the ratio between 10 degrees and 20 degrees in respiration (RQ), the weight of oxygen used by a 1g fish at $0\deg C$ while not swimming (RA) and finally, the coefficient for the allometric mass function (RB). Similar to C_{max} of the *consumption* function, an R_{max} is calculated. It is then multiplied with a factor (R) determined by all other parameters, which is then multiplied again by an optional activity factor ($ACTIVITY$) to calculate the final respiration value.

$$\begin{aligned}
RY &= \log RQ \cdot (RTM - RTO + 2) \\
RZ &= \log RQ \cdot (RTM - RTO) \\
RX &= \frac{RZ^2 \cdot (1 + \sqrt{1 + \frac{40}{RY}})^2}{400} \\
V &= \frac{RTM - T}{RTM - RTO} \\
R &= \begin{cases} V^{RX} \cdot e^{RX \cdot (1 - V)} & \\ 0.000001 & ifT >= RTMorR < 0 \end{cases} \\
R_{max} &= RA \cdot W^{RB} \\
respiration &= R_{max} \cdot R \cdot ACTIVITY \cdot OXYCAL
\end{aligned} \tag{8}$$

The *egestion* function is depicted in Equation 9. This is the equation proposed by Stewart, Weininger, Rottiers, *et al.* It is parameterized by the max amount of food that can be consumed (C), the current temperature (T) and the *p-value* (p) from previous computations. As a set of model parameters, it uses egestion in grams per day (FA), the coefficient expressing the dependency between water temperature and egestion (FB), and the coefficient for the dependency between the p-value and egestion (FG). This function also takes into account the density of the prey, with support for multiple prey types and their energy density (PFF).

$$\begin{aligned} PE &= FA \cdot T^{FB} \cdot e^{FG \cdot p} \\ PFF &= \sum_{i=1}^{n_prey} prey_i \cdot proportion_i \\ PF &= \frac{PE - 0.1}{0.9} \cdot (1 - PFF) + PFF \\ egestion &= PF \cdot consumption_{max} \end{aligned} \tag{9}$$

Finally, the *excretion* function[15] used is depicted in Equation 10. It uses the temperature (T) and the previously computed *p-value* (p), maximum consumption ($consumption_{max}$) and egestion ($egestion$). It is then parameterized using an excretion coefficient (UA), a coefficient for the dependency between water temperature and excretion (UB), and a coefficient for the dependency between the *p-value* and excretion (UG).

$$excretion = UA \cdot T^{UB} \cdot e^{UG \cdot p} \cdot (consumption_{max} - egestion) \tag{10}$$

Parameter Estimation The estimation of parameters for bioenergetics models, such as the Wisconsin fish model, is a crucial step in accurately predicting fish energy consumption. These parameters are often species-specific and can vary based on factors such as temperature, food availability, and fish size. Estimation of these parameters is typically done through experimentation in real-life environments, such as lakes or fish hatcheries. Researchers can monitor the growth and energy consumption of fish in these environments while simultaneously measuring relevant variables such as water temperature and food availability. This information is then used to estimate the parameters for the bioenergetics models, which can be applied to predict energy consumption in a variety of scenarios. Accurately estimating these parameters is essential for effective fisheries management and conservation efforts.

4.2 Fish Behaviour

Although fish are relatively simple creatures they can, like many animals in the wild exhibit complex behaviour when placed in a group. Birds fly in flocks and their distinct V-shaped formation to save energy. Ants and bees work together to build complex structures and find food. Fish also exhibit complex behaviour when placed in a group, and this behaviour is often referred to as schooling. Some key advantages for the fish to have this schooling behaviour are that they can save energy by swimming in the wake of other fish, and they also have a better chance at survival when attacked by a predator. Furthermore, it also helps the school forage for food more efficiently. It is, however, difficult to capture these behaviours in a computer simulation. The following section will describe some of the approaches that have been taken to simulate this behaviour.

Swarming A popular approach to this problem, first introduced by Huth and Wissel in 1992[22] is to calculate the next location of an individual based on its neighbours. This is achieved by combining several rules, which are applied to each individual in the swarm. The original terms to describe this behaviour were: repulsion, attraction and searching. These terms have since been replaced by more modern terms such as separation, cohesion and alignment. These terms are also commonly used to describe the behaviour of other swarm species such as birds and bees.

Perception According to Huth and Wissel[22] the introduction of a perception range and angle improves the accuracy of the simulation. For a given fish, any of its neighbours is in range, and thus in the fish's perception range if the distance between the two fish is less than the perception range, and the angle between the two fish is less than the perception angle. Both the perception range and angle are configurable parameters. They also propose to use different range parameters for the different components of the behaviour.

Cohesion is the tendency for the fish to form a group and stick together. It is calculated by taking the average position of all the fish in the perception range and subtracting the current position of the fish. This results in a vector that points towards the average position of the fish in the perception range. It is defined by Equation 11.

$$\vec{C} = \frac{\sum_{i=1}^{n} \vec{P_i} - \vec{P}}{n} \tag{11}$$

Separation is the tendency for the fish to avoid crowding each other. This term is the negative of the cohesion vector, but due to different ranges for these terms, it is required to recalculate this term. It is defined by Equation 12.

$$\vec{S} = \frac{\sum_{i=1}^{n} \vec{P} - \vec{P_i}}{n} \tag{12}$$

Alignment is the tendency for the fish to move in the same direction as their neighbours. It is calculated by taking the average velocity of all the fish in the perception range. This results in a vector that points in the same direction as the average velocity of the fish in the perception range. It is defined by Equation 13.

$$\vec{A} = \frac{\sum_{i=1}^{n} \vec{V_i}}{n} \tag{13}$$

Computing the *preferred velocity* [22] is done by summing up the different terms and multiplying them by their respective weights. This velocity is then multiplied by the maximum velocity of the fish, and the timestep over which it is calculated. The preferred velocity is defined by Equation 14.

$$\vec{V_p} = w_c\vec{C} + w_s\vec{S} + w_a\vec{A} \tag{14}$$

4.3 Environment

The environment that the fish are simulated in affects both the energy consumption of the fish as well as the behaviour of the fish. Bioenergetics models are often parameterized by temperature, e.g. the respiration component of the Wisconsin Fish Model [21]. Temperature also affects the food consumption rate for most species [20], where they stop consuming food at a certain cut-off temperature, effectively killing them.

The behaviour of the fish is also influenced by the environment, as the availability of food and the presence of predators can influence the behaviour of the fish. The presence of predators can cause the fish to move erratically, as they try to avoid the predators. The presence of food can cause the fish to move towards the food, or to stay in a certain area for a longer period in order to feed on it.

The simulator will have a static temperature, although a temperature gradient can easily be implemented, as well as a small set of randomly placed food sources. The exact configuration of these will be detailed in the experiments section.

5 Implementation

FishEcoModeler has been implemented in Unity, which provides a stable and easy-to-use environment for running scripts. Unity supports multiple programming languages, with C# being the default configuration.

Although Unity is not known for its high performance in simulation, it is sufficient for this thesis. The primary reason for choosing Unity is its extensive documentation, which facilitates the integration of other frameworks and libraries. This is essential as the framework is intended for use by other researchers and should be user-friendly.

If performance issues arise, it may be necessary to port the framework, or certain components of it, to a different framework such as MuJoCo [48]. However, this would require significant effort since the current implementation is closely tied to Unity.

5.1 Architecture

FishEcoModeler is designed to be as flexible as possible, which means that it can be used for a wide variety of experiments. This is accomplished by using a modular design, which allows for easy extension of the framework.

Out-of-the-box Unity uses the GameObject architecture, which is a hierarchical system where each GameObject can have a set of components (scripts). These components are used to store data and implement functionality. For instance, a GameObject can have a *Rigidbody* component, which is used to store the object's mass and velocity. The GameObject system is great when working with a relatively small, but complex set of objects. However, it does not scale well when working with a large number of objects, and it does not take advantage of the parallelization capabilities of modern CPUs. An example of this architecture can be seen in Figure 1.

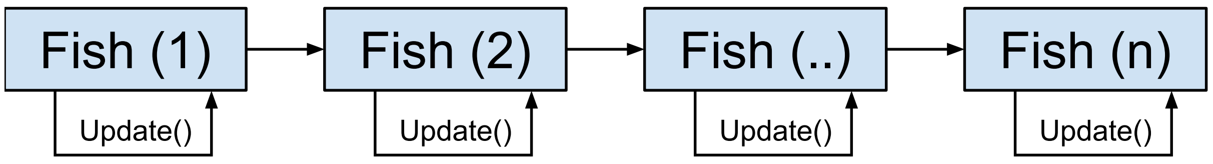

Figure 1: *The traditional GameObject-based architecture, GameObject or fish are updated in sequence, when the GameObject updates, all of its components are updated. GameObjects can be complex objects and memory isn't always contiguous.*

Performance As mentioned previously, the GameObject architecture does not take advantage of multiple threads to update the state of the simulator. Which in turn makes it very computationally expensive to run the simulation. Early experiments showed that the simulation would run at 5 frames per second with only 250 fish. This means that the simulation would run at the same speed as real-time, thus making it unsuitable to be used as a predictive tool.

In order to improve performance, it is necessary to parallelize the simulation, as is commonly done for Particle Swarm Optimization algorithms [38] This is a non-trivial task, as Unity's support for parallelization is limited. In 2019, Unity introduced the DOTS[5] platform. Most of the packages for this platform are still in preview and evolve rapidly. With this platform, Unity moved away from the GameObject architecture and brought the well-known *Entity-Component-System* (ECS) architecture into Unity.

Entity-Component-System (ECS) is an alternative to the GameObject architecture. And as the name suggests consists of three different parts: *entities*, *components*, and *systems*. Entities represent objects within the architecture, these objects then contain multiple components which are used to store data. Systems implement the functionality, manipulating the entities based on the data components that it has. It is this manipulation of a batch of entities that allows for parallelization.

DOTS is Unity's implementation of the ECS architecture. Using it has multiple advantages over the GameObject architecture;

1. **Parallelization** - DOTS automatically handles the creation and management of threads and allows users to run small bits of code also known as *jobs* which can be run in parallel.
2. **Data locality** - DOTS automatically groups entities in memory based on their set of components, which with data locality, in turn, improves performance.
3. **Unity's Burst compiler** - a compiler for the .NET intermediate language (IL) to highly-optimized machine code. This allows for significant performance improvements.
4. **Extensibility** - Due to the ECS architecture, DOTS is incredibly modular, the addition of new functionality can be done by adding new systems, requiring minimal changes to the existing code.

Figure 2 shows a total of three systems, System A and System B inherently have a dependency on each other as they access the same entities. Their application logic can however be split, e.g. System A could be responsible for updating the position of the fish, while System B could be responsible for updating the energy level of the fish - this makes it easy to modify specific features of the simulator. System C is independent of the other two systems and could for example be responsible for regrowing the food sources. If new features were to be added to the simulator, it would simply be done by adding a new system, which can make use of existing components without having to modify the existing code.

Figure 1 shows the traditional GameObject. The major difference between the two is that in

[5]https://unity.com/dots

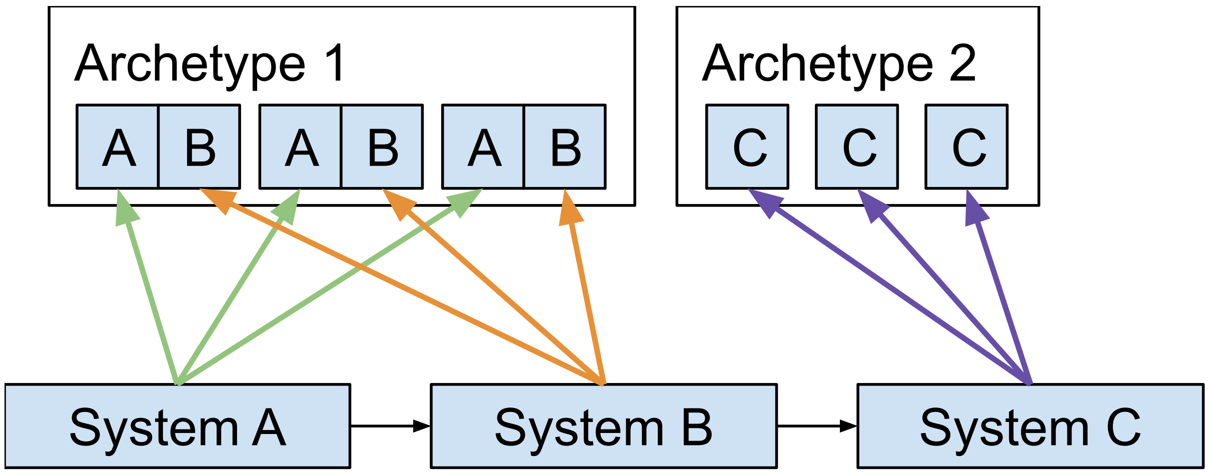

Figure 2: *The ECS architecture, Entities, denoted by a combination of components (A/B), are updated by systems that specifically target components. Entities are grouped by DOTS based on their components (Archetypes), which allows for better memory locality. This pattern also allows batching of entity manipulation, denoted by the arrows, which can be used for multi-threading.*

the GameObject architecture, every GameObject is updated sequentially, while in the ECS architecture, entities are grouped by their components, which allows for better memory locality and parallelization, systems then target components specifically and update all entities that have those components.

This does not only performance but also brings some interesting features, Unity implemented all major components of the engine using DOTS, which means that built-in features such as physics and rendering can be customized and extended. This makes it trivial to for example tie the framerate to the physics simulation by overriding the built-in `RateManager`, something which is significantly more difficult in the GameObject architecture.

5.2 Features

In order to achieve the modular design goal, all different modules of *FishEcoModeler* are implemented as *systems*, which are run sequentially by the Unity DOTS framework, with the jobs in the performance-critical systems being run in parallel.

Fish Behaviour The system that governs fish movement is the *FishbehaviourSystem*, which given the current position, calculates the next position for each fish. This is accomplished by applying the formulas described in Section 4.2. Like Particle Swarm Optimization, the system benefits significantly from parallelization[38].

Fish Bioenergetics Fish bioenergetics models are implemented in the *EnergySystem* which for every fish tracks and updates its energy level. Combining this with the 'FishBehaviourSystem' forms the basis of *FishEcoModeler*. This system is also responsible for removing dead fish from the simulation.

Environment The environment is not defined by a single system, but rather a collection of systems. One example of such a system is the *FoodSourceSystem*, which is responsible for spawning and updating food sources. Another example would be the *TemperatureSystem*, which provides the temperature at a given location or changes it over time with day and night cycles. Other examples include the *ReproductionSystem* which manages the reproduction of fish, when a certain threshold is met, effectively creating a clone of the current fish, or something which could be further extended into creating fish spawn which is then handled by another system.

Analytics The *StatisticsSystem* is an example of another system within *FishEcoModeler*. This system does not influence the simulation and is not run as frequently as other systems. Its purpose is to gather statistics about the simulation, such as the average energy level of the fish or the total amount of food available, which are then saved to a file.

While running the simulation, the *StatisticsSystem* periodically records data about the fish population, such as their energy levels and the available food in the environment. This data is then written into a file which is processed and visualized using Python scripts. The Python scripts provide an easy-to-use interface for processing and graphing the data, which can be customized for the user's needs. The ability to visualize the data in this way allows for a better understanding of the fish population dynamics and the impact of different factors on the population's behaviour.

Configuration Configuration of *FishEcoModeler* is done through a *JSON* file. This file contains all the parameters that are used by the framework, such as the number of fish, the size of the environment, and the initial energy level of the fish. The file is parsed and the parameters are used to initialize the simulation. This approach allows for easy configuration of the framework and makes it easy to share and reuse configurations.

Visualization Figure 3 shows the architecture as implemented in Unity DOTS. Some systems like the *ProducerSystem* do not have any jobs linked to them, this is because they directly manipulate the components from the system update method for which DOTS uses code-generation to create jobs. Another example is the Configuration entity, which stores multiple components which are excluded for brevity. Dependencies for the *StatisticsSystem* are also not shown, as it is directly linked to all components presented in the figure. The fish and food sources also contain a set of Unity physics-related components.

5.3 Benchmarking

In order to assess the performance of *FishEcoModeler*, a series of benchmarks were run. Benchmarking the simulator allows other researchers to estimate the viability of the framework for their experiments. For example, a research project studying ecological impact over

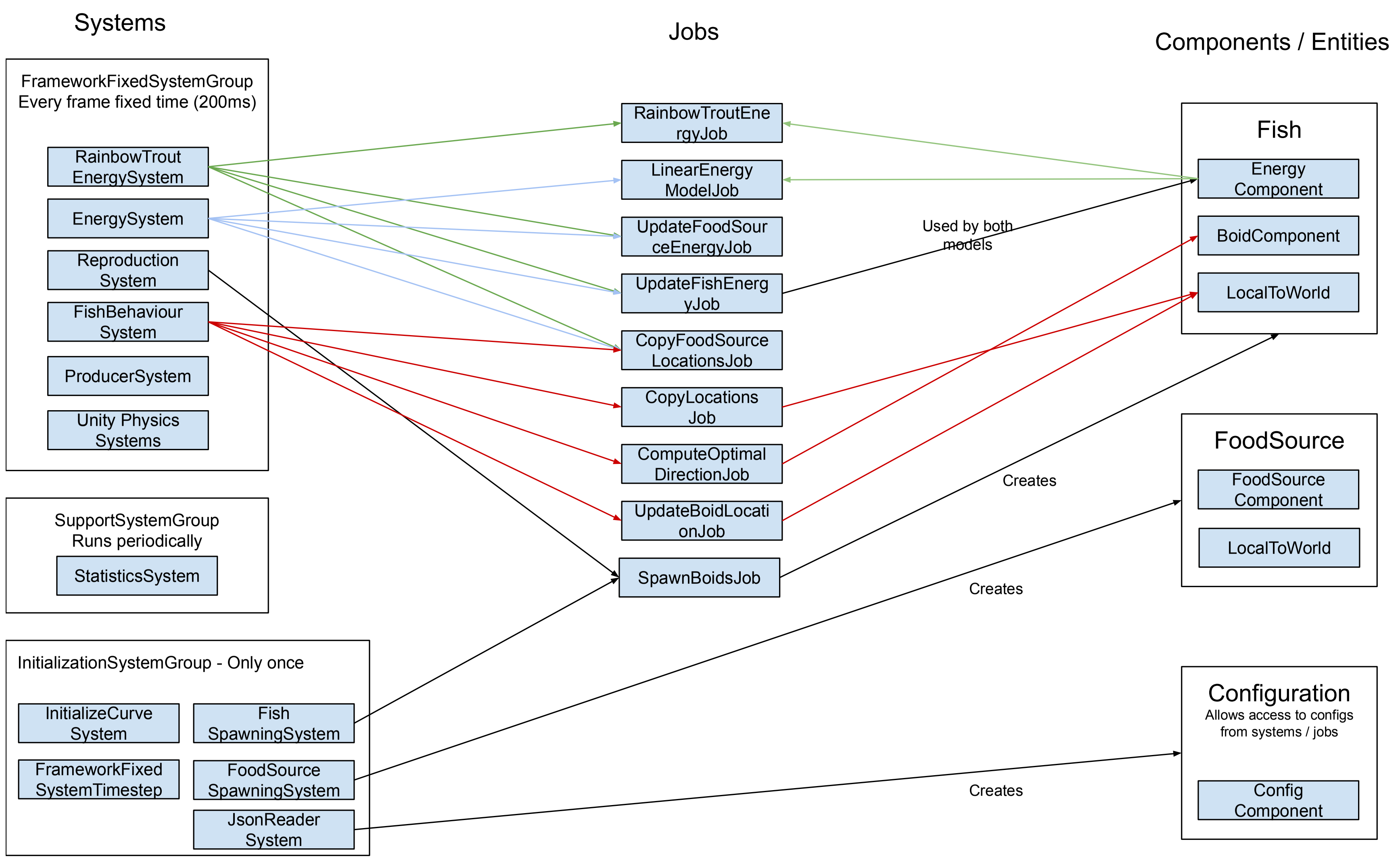

Figure 3: *Architecture of the FishEcoModeler framework. It is built on top of Unity DOTS, which allows for parallelization of the simulation. The systems use (parallel) jobs to update the state of the componentes inside the entities. Note that this is a simplified version of the architecture, Some systems modify components directly, and these jobs are generated by DOTS.*

10 years would require a different performance profile than a research project studying the impact of a single event and its short-term (10 weeks) impact on the ecosystem.

Setup The benchmarking was done on a computer with the following specifications:

- Processor: AMD Ryzen 7 5800X 16-Core Processor
- Memory: 32GB DDR4
- Graphics Card: NVIDIA GeForce RTX 3070

This computer as of 2023 is considered a high-end consumer computer, it is also important to note that the graphics card is not used by the simulator, as the simulation is run in batch mode, without any graphics. Although the simulator updates the fish in parallel, it benefits significantly from having a processor with fast individual cores, especially with lower fish counts (< 25). As for memory, the simulator rarely uses more than 200MB of memory, although this is expected to increase with the addition of more complex systems.

The simulator is built from the Unity Editor and then ran with the `-batchmode` and `-nographics` flags, these flags prevent visuals from being rendered and allow the simulator to run on headless machines. The simulator runs a specific configuration file, which is detailed in the next sections. The time it takes to simulate a single hour is then measured and recorded by the pre-existing *StatisticsSystem*. A total of 5 days are simulated, and the average time per hour is then computed.

In general, the configuration has the following key properties;

- **Fish count** - The number of fish in the simulation, variable.
- **Food count** - The number of food sources in the simulation, variable.
- **Updates Per Second** - The number of times the simulation is updated per second, for these experiments this parameter is set to 5.

Benchmark: Fish Movement As explained in section 3 the simulation of fish movement has a nearly quadratic complexity of $O(n \cdot (n-1))$, therefore the expected performance of the simulator is heavily dependent on the number of fish in the simulation. This will therefore be the first benchmark that will be run. The expected slope is $n \cdot (n-1)$. In order to translate this to seconds, the difference between the time it takes to simulate 1 hour with 1 fish (ft_1) and 5 fish (ft_5) is calculated, this difference is then simply divided by 4 to get the time it takes to simulate a single fish (f_c). The n in the previous formula is then scaled with this scalar and the constant $overhead = ft_1 - f_c$ is added to account for the overhead of the simulator.

Figure 4 shows the results of this benchmark. It is observed that as the number of fish increases the time it takes to simulate an hour increases. The slope of the graph is nearly linear and with the slope only increasing slightly, but not with the exponential as expected.

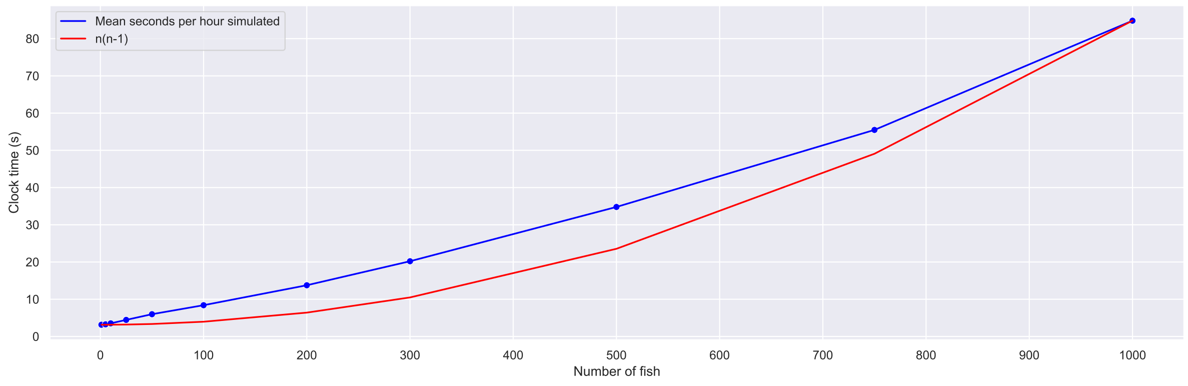

Figure 4: *Running time proportional to the number of fish being simulated.*

An observation made during the benchmarks was that the simulator was unable to utilize all 16 cores, which was only observed with fish counts > 750. Upon closer inspection, it is revealed that a significant portion of every frame is spent on allocating memory for the positional information of the fish to be shared with the neighbours, this operation is expected to linearly increase with the number of fish.

As much as DOTS tries to divide the load over different threads, some processes still run on the main thread, such as sync points after systems run and large memory allocations. In normal operation for DOTS, this would not be an issue, as the systems would not be run at such a high frequency as they do for the simulator, thus cutting down on the overall combined overhead on the main thread. In the case of the simulator however, the systems run at a very high frequency of approximately $5707hz$ for a single fish, with the rate decaying to $2142hz$, $517hz$ and $212hz$ for 100, 500 and 1000 fish respectively. This means that performance is not limited by the number of fish, but rather by processes that run on the main thread.

Benchmark: Fish Bioenergetics Model One of the primary advantages of the simulator is that it works with existing bioenergetics models. In the past, these models were considered to be too complex for simulation [6] this is understandable, as these models are complex mathematical equations. In order to test this (old) claim an experiment is executed with the Rainbow Trout bioenergetics model from *FB4* (detailed in section 4), and the wall time is measured while the simulator is running.

It is worth noting that the bioenergetics model is running on the main thread only, as it is also responsible for updating the food sources with the amount of food that is consumed by the fish. During the benchmark, similar problems regarding CPU utilisation were observed, where the simulator was not able to fully utilize all 16 cores of the CPU. These problems started occurring when simulating > 300 fish.

Figure 5 shows the results of this benchmark. It is observed that as the number of fish increases the time it takes to simulate an hour increases. This plot shows both the wall

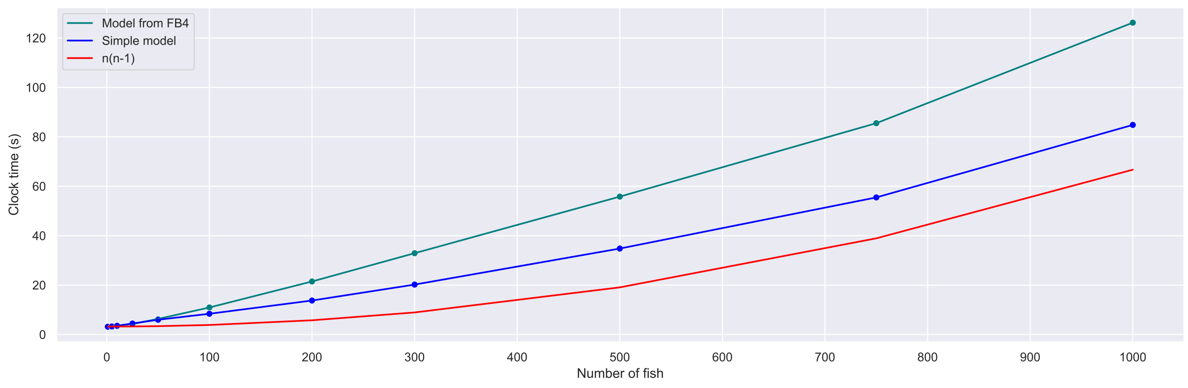

Figure 5: *Running time proportional to the number of fish being simulated with a more complex energy model.*

time it takes to simulate an hour, as well as the predicted time if the simulator would have an $O(n \cdot (n-1))$ time complexity, with the scalars computed using the same method as in the previous benchmark. The results of the previous benchmark are also included in this plot for comparison.

This figure shows that the line deviates even more from the expected profile, which highlights the impact that the requirement for the bioenergetics model to run on the main thread has on the performance of the simulator. Even though this is not the expected performance profile, the simulator still performs relatively well, with it being capable of simulating a day for a school of 1000 fish in less than a single hour.

6 Experiments & Results

To assess the capabilities of *FishEcoModeler*, a series of experiments will be carried out, with each focusing on a specific aspect of the framework. These experiments will provide valuable insights into the functionality and limits of the framework and also serve as a showcase of the framework's capabilities. The experiments are therefore divided into three different categories.

Validating the Accuracy and Reliability of the Simulation Framework These experiments are designed to establish the accuracy and reliability *FishEcoModeler* in modelling ecosystem behaviour. Through monitoring the average energy level of the fish population and the grammage available in the simulated environment, the goal is to demonstrate that the framework behaves like any natural ecosystem. It is expected that elements such as population cycles[49] are observed in the results of these experiments.

Validating the simulation framework in this manner will provide confidence in the results of future experiments that use the framework to study fish energy consumption in diverse scenarios.

Fish Energy Consumption Modelling In this class of experiments, the energy consumption of fish will be studied by isolating it from other ecosystem variables. This can be accomplished by for example configuring the food sources in such a way that they do not deplete over time. This approach will enable us to evaluate the energy consumption of fish and compare the results with the commonly used program, Fish Bioenergetics 4 (FB4) [15].

By comparing the results of this experiment with those obtained using *FB4*, the experiment will validate the simulation framework's ability to accurately model fish energy consumption. As the methods used in *FB4* are well-established and widely studied, any discrepancies or differences between the two sets of results can provide valuable insights into potential areas of improvement for the simulation framework.

Modelling Complex Environmental Properties This class of experiments aims to highlight the potential and possibilities of the simulation framework by introducing complex (spatial) environmental properties such as differences in temperature or oxygen which are related to the depth of the water. These experiments are mostly a demonstration of the framework's capabilities and when properly combined with existing bioenergetics models can be used to study the impact of these properties on fish energy consumption.

6.1 Experimental Setup and Parameters

Parameters are stored in a file with the *JSON* format. This file can be consumed by the framework as is detailed in 5.2. Every experiment has its configuration file, which can be found in the appendix.

Fish Parameters The experiments will be run with the intention of simulating rainbow trout in a lake-like environment. A speed of $0.84\,\mathrm{m\,s^{-1}}$ [5] is picked, as the first experiment is run with a simple linear energy model, A linear coefficient is computed using the data from Pottinger, Rand-Weaver, and Sumpter [39] where they observed multiple groups of rainbow trout during 120 days. The groups started with an average weight of 283g and at the end of the experiment, the average weight for the fed fish was 530g whereas the fasted fish only weighed 219 grams.

The gain of the fed fish per day can then be computed as follows: $(530 - 283)/120 = 2.06$ the loss of the fasted fish per day is also computed: $(219 - 283)/120 = -0.56$. This is then normalized to seconds for use in the framework and it is used for the *ConsumptionRate*.

To calculate the feeding rate it is assumed that the fed fish lose as much weight as the fasted fish group, offset by the gain of the fed fish group. This results in a linear coefficient of 2.59 grams per day. This is used as the *FeedingRate* for the next experiments.

Compute The experiments are run on the same computer as the one used in section 5.3, the simulator runs as fast as theoretically possible, due to the modified system rate manager. As the simulation is deterministic, and all statistics are aggregates, experiments are only run a single time.

6.2 Validation of Simulator Precision

The objective of this experiment is to validate the internals of the simulator, as the energy consumption of a fish per day is only 0.56 grams, in order to use this value for simulation it has to be normalized to seconds, and the simulator itself will then divide it further to a timestep. This means that the simulator has to be able to handle very small values such as $0.56/(24 \cdot 60 \cdot 60 \cdot 5) \approx 1.30e - 6$. This experiment should show that the simulator is implemented with sufficient precision to handle these values.

Hypothesis It is expected that the simulator is able to handle these values, as the framework is implemented in C# which has multiple data types that can handle these values. The simulator internally uses the *decimal* type which has a precision of 28-29 digits and according to Microsoft intended for financial applications[6]. It is expected that the average weight of the fish will linearly decrease over time with a slope of -0.56 grams per day.

Setup For this experiment no food sources will be present and the fish will be configured to have a linear energy consumption model. The starting weight of the fish is set to 283 grams, which is the average weight of the fish in the experiment by Pottinger, Rand-Weaver, and Sumpter and was also used to compute the linear coefficient. The fish will be simulated for 30 days and the simulator will update 5 times per second. They will consume their

[6]https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/floating-point-numeric-types

energy at a rate of $0.56/(24 \cdot 60 \cdot 60) = 6.48e - 6$ grams per second. This experiment is run with a small group of 25 fish.

Results & Dicussion Figure 6 shows the results of the experiment. It is observed that the average weight of the fish decreases linearly with a slope of -0.56 grams per day. This confirms our hypothesis that the simulator can accurately process the energy consumption of the fish.

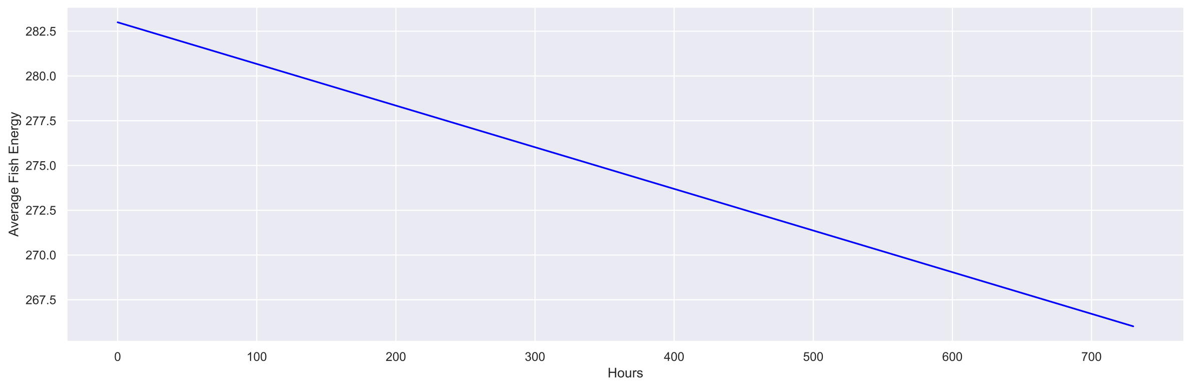

Figure 6: *Results of experiment 1. This figure shows the average energy of the fish over time. The energy decreases linearly with a slope of -0.56 grams per day.*

Whilst carrying out this experiment, multiple issues were discovered with the simulator, during development, arbitrary values were used to roughly test the working of the simulator, with most fish often having a growth rate of $0.1g/s$ which would result in the fish gaining almost 9 kilograms per day. When transitioning to realistic values such as $0.56g/day$ it was discovered that the simulator was not able to handle these values, as the fish would not lose any weight. It was later discovered that this was due to a precision problem, as the standard *float* type in C# only has a precision of 7 digits. Swapping this over to *double* for configuration and *decimal* for the simulator internals fixed this issue.

Conclusion Although simple, this experiment was extremely valuable. It exposed a major issue with the simulator and allowed for premature validation of its internals. It also served to verify the linear energy consumption model, which is used in subsequent experiments.

6.3 Validation of Environmental Precision

The objective of this experiment is similar to the previous experiment, but instead of validating the internal fish model, an environmental system is validated, in this case, the *ProducerSystem* as detailed in Section 5.1. When implementing environmental properties, they will be implemented in a similar way to this system. With this experiment, the aim is to show that this aspect of the framework is implemented with sufficient precision to handle the values that will be used in the simulation.

Hypothesis Similar to the previous experiment, the simulator is expected to be capable of handling these values, as the *ProducerSystem* internally also uses the *decimal* type. Without any fish, The food source is expected to grow linearly with a slope of 65.416 grams per day which would be enough to feed a small group of 25 fish.

Setup For this experiment no fish will be present and the environment will only have a single food source with a starting weight of 0. The food source will grow linearly, and the maximum size of the food source will not be capped. The growth rate of the food source is set to 65.416 grams per day, or $65.416/(24 \cdot 60 \cdot 60) \approx 7.57e-4$ grams per second. The food source will be simulated for 30 days with the simulator updating 5 times per second.

Results & Dicussion Figure 7 shows the results of the experiment. It is observed that the average weight of the food source increases linearly with a slope of 65.416 grams per day, which confirms the hypothesis that the simulator can accurately process the growth of the food source.

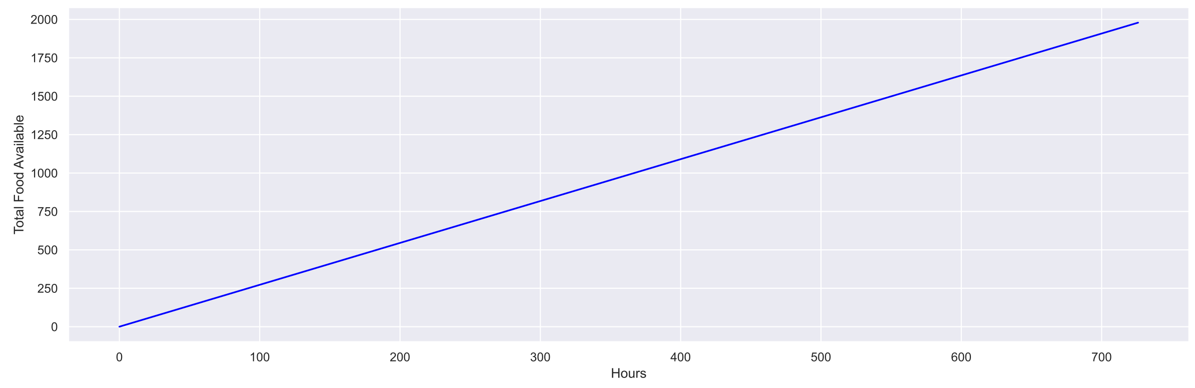

Figure 7: *Results of experiment 2, the figure shows the total food available in the environment over time. The food sources grow linearly with a slope of* 65.416 *grams per day.*

These results were again only obtainable by fixing a precision issue, similar to the one in experiment 1. The *ProducerSystem* was already using the *decimal* type, but due to compatibility issues with Unity's built-in JSON parser, this was not properly converted. This was fixed by simply changing the configuration type to *double* and converting it to *decimal* when the simulation starts, as the extra precision is only needed during the simulation.

Conclusion Although experiment 2 was low in complexity, comparable to experiment 1, it again was valuable because it again showed the importance of the precision of the simulator. It also serves as validation of the *ProducerSystem* and the way environmental properties will be implemented in the framework.

6.4 Basic Environment Interaction Between Fish and Food Sources

This third experiment, like the previous two, also falls into the validation category. With this experiment, the goal is to validate the interaction between the fish and the environment, more specifically, the average fish weight in comparison to the food available.

Hypothesis It is expected that the mean weight of the fish will linearly increase, with the slope of this line being 2.06 grams per day. This is a combination of the feeding rate and consumption rates computed in 6.1. It is also expected that the total weight of food available in the environment stays stable, as the fish will consume the food at the same rate as it is produced.

Setup The experiment is run for 30 days, with a small group of 25 fish. For the consumption parameter, the same value from the first experiment is used; $6.48e-6$ grams per second. There will only be a single food source in the environment, with a starting weight of 450 and a growth rate of $7.57e-4g/s$. The fish will be able to feed themselves at a rate of $3.03e-5g/s$.

Results & Dicussion Figure 8 shows both the weight of the fish increasing over time as well as the total weight of food available in the environment. It is observed that the average weight of the fish increases linearly, but not with the expected coefficient. The slope of the line is 1.27 grams per day, which is significantly lower than the expected 2.06 grams per day. It is also observed that the total weight of the food source increases over time - 19.82 grams per day, which is not expected.

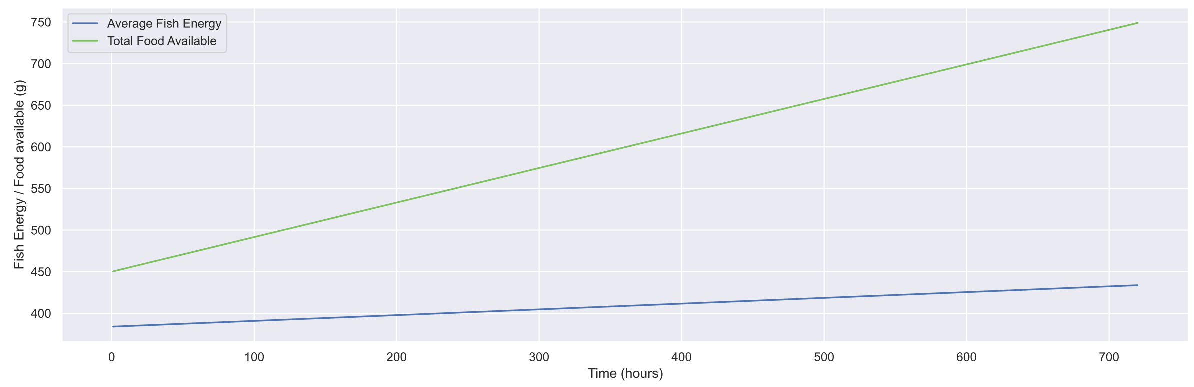

Figure 8: *Results of the first set of parameters for experiment 3 - a linear increase in the average weight of the fish is observed, but not with the expected coefficient. This figure also shows the total weight of the food source increasing over time, which is unexpected.*

Empirically it was discovered that this is due to the *FeedingRadius* configured on the food source. As mentioned in 5.2, the fish will only consume food if they are within a certain radius of the food source. It is theorized that due to the schooling behaviour of the fish, a portion of the fish will always be outside of this radius, which means that the food will not be consumed by all 25 fish concurrently.

Figure 9 When the experiment was re-run with a *FeedingRadius* of 10, it is observed that the total weight of the food source stays stable, as expected. The average weight of the fish also increases linearly, with a slope of 2.06 grams per day, which is the expected value.

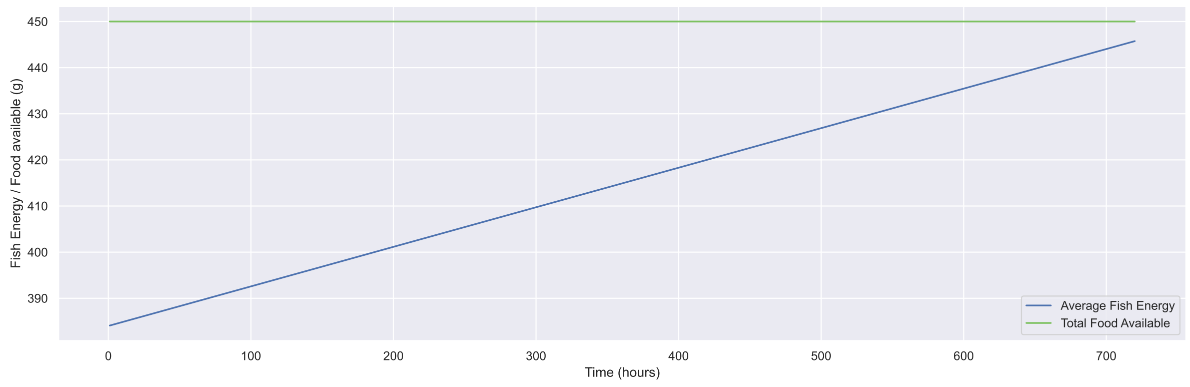

Figure 9: *Results of experiment 3 when ran with a feeding range of 10. The expected results are observed, with the Total Food Available remaining stable and the average weight of the fish increasing linearly with the expected slope of 2.06 grams per day.*

Although tweaking the *FeedingRadius* parameter made the simulator behave as expected, it is not a realistic solution. In a real-world scenario, the fish would only be able to eat when relatively close to the food source, making 5 meters unrealistic. In most experiments with real fish in current literature, the fish are fed by spreading the food over the entire surface of the water, effectively ensuring that the fish always have access to food near them[39].

There are multiple solutions to this problem, one being to implement a specific feeding behaviour, where the fish will actively seek out food sources. Another solution is to increase the *FeedingRate* to account for the fact that the fish will not always be able to eat. This is the solution that was chosen for this experiment, as it is the easiest to implement and does not require any changes to the existing fish behaviour.

Given the coefficient of the experiment with feeding range 1, c_1, and the coefficient of the experiment with feeding range 10, c_10, the feeding rate fr and consumption rate cr the new feeding rate can be computed using Equation 15.

$$fr_{adjusted} = fr \cdot \frac{c_1 + cr}{c_{10} + cr} \tag{15}$$

Plugging in the values used in the experiments results in a required scalar of 1.179 which results in an adjusted feeding rate of $3.57e-05$ grams per second. Simulating with these values results in a graph visually equal to Figure 9, which is the expected result. An interesting detail to this is that small fluctuations in the available food can still be observed, which is a result of the simulated movement of the fish. This is highlighted in Figure 10.

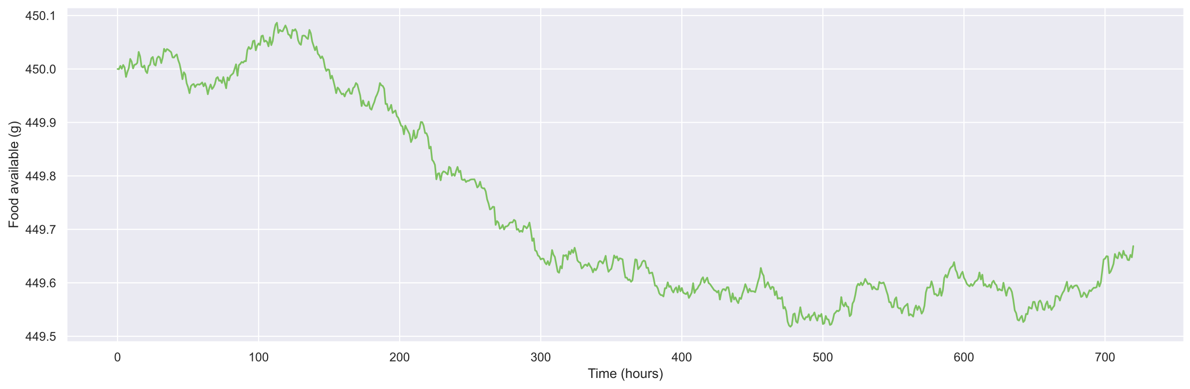

Figure 10: *Minor fluctuations in the total weight of food available to the school in grams over time. In this graph, although only visible in minor fluctuations, the total weight of food available to the school is not completely stable, this is due to the simulated movement of the fish.*

Conclusion Although the initial results of this experiment were not as hypothesised, the cause of these inconsistencies was identified, which highlights the differences in complexity between this simulator and existing solutions. Changing the parameters, and making assumptions about the behaviour of the fish, resulted in the expected results.

This experiment also showed some considerations when using the simulator, such as whether or not a more realistic feeding behaviour should be implemented, or if the current, where the fish absorb nearby food, is sufficient. Although out of scope for this thesis, it is an interesting question to consider for future work.

6.5 Rainbow Trout Bioenergetics

Since *FB4*[15] is one of the main inspirations for this simulator, it was decided that for this experiment a bioenergetics model from *FB4* will be implemented as a system in the simulator, and then compared against the results of *FB4*. The aim is to show that the simulator is capable of accurately simulating energy consumption using an existing model.

Hypothesis As shown in the previous experiment, the simulator, given minor parameter adjustments to account for the simulation of the fish their behaviour, accurately simulates the energy consumption of a fish with a linear energy model. Although the models present in *FB4* are significantly more complex, It is expected that the simulator can accurately simulate the energy consumption of a fish using the *FB4* model.

Setup For this experiment, the same parameters as in the third iteration of the previous experiment are used. Instead of using the linear model, however, a new type of model is implemented, stitching together functions of the existing *FB4* software. The model is implemented as a separate system in the simulator, making it easy to toggle on and off. As stated in section 5.2 This model is implemented as a separate system, and a simple toggle is built in to switch between a linear model and the *FB4* model.

The rainbow trout model is an extension of the Winconsin model, detailed in section 4.1. As the original model is designed for daily predictions, and not 5 predictions per second, the new fish weight is calculated as if it has grown for a day, then compute the difference between the predicted weight and the current weight, this is then scaled to a singular simulator step and apply it to the current weight.

The model is built for a constant supply of food, similar to the previous experiment. The *Ration_prey* parameter can be used to control the intake of the fish, so as an extension of the model, it is replaced with a function that either takes the configured *FeedingRate* when there is a food source nearby, or 0 when there is not. *FeedingRate* is calculated by the number of grams that the fish are fed per day, divided by the number of seconds in a day, which is then multiplied by the same magic constant from the previous experiment to account for the schooling behaviour of the fish.

In order to test the hypothesis three different sets of parameters are tested. In the first experiment, the *Ration_Prey* parameter is fixed to $5.79e-5$ or 5 grams per day. It is then fixed to 0 in order to simulate an environment without any food. Finally, it is set to $6.83e-5$ or 5.90 grams per day, with the special *Ration_prey* function enabled, this would be the intended use-case for the simulator.

Results & Dicussion Figures 11 and 12 show the results of the configurations where *Ration_prey* is fixed to 5 and 0 respectively. Both the results from the simulator and *FB4* with the same parameters are displayed. *FB4* supports a min timestep of 1 day, which is represented using dots in the figure.

It is observed that the results from the simulators are equal to those from *FB4*, further increasing confidence in the simulator's internal working.

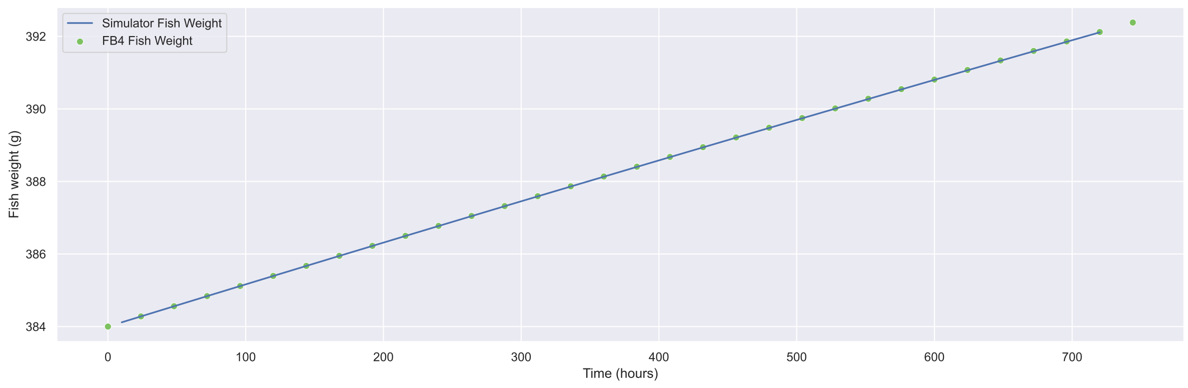

Figure 11: *Weight of the school of fish (g) increasing over time when the Ration_prey parameter is fixed to* 5 *grams per day. The results from the simulator are equal to those from FB4.*

Figure 13 shows the results of the configuration where *Ration_prey* is set to 5.90 grams per day, with the special *Ration_prey* function enabled. Significant deviations are observed after a week of simulation, with the simulator predicting a lower weight ($-0.3g$) than *FB4*.

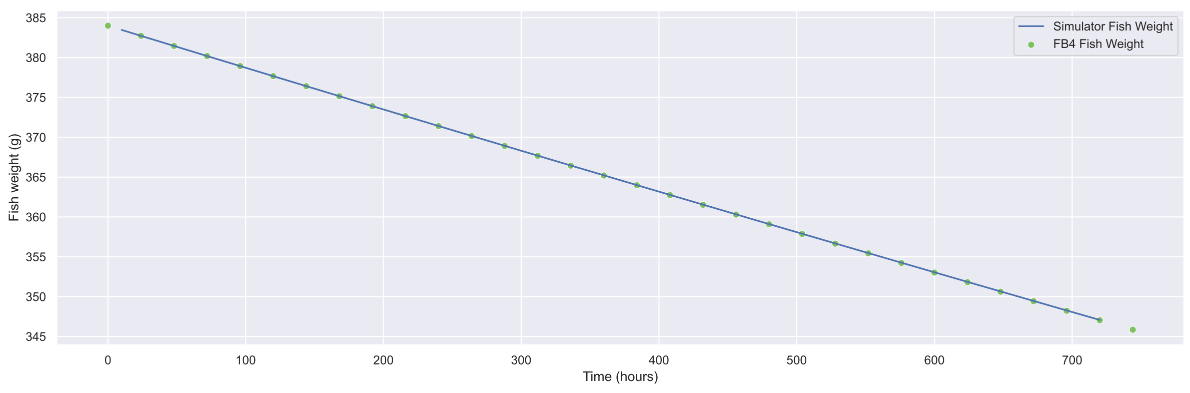

Figure 12: *Weight of the school of fish (g) increasing over time when the Ration_prey parameter is fixed to* 0 *grams per day. The results from the simulator are equal to those from FB4.*

When computed, the slopes for the simulator and *FB4* are 1.21 and 1.27 grams per day respectively. It is hypothesised that this has a similar cause to the previous experiment, where the fish are not always able to eat due to the *FeedingRadius* parameter.

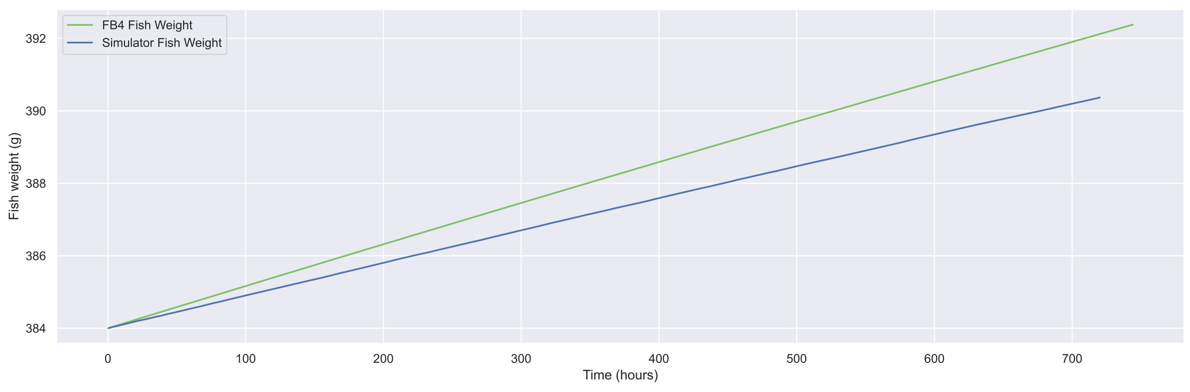

Figure 13: *Weight of the school of fish (g) increasing over time. This is the first configuration where significant differences from the FB4 results are shown.*

Similar to Figure 10 from experiment 3, small fluctuations are also present in the weight of the individuals in the school, see Figure 14. Very minor non-significant fluctuations in the weight of the fish are observed in a single day. This is consistent with the conclusion drawn in the previous experiment where the fish movement influences the amount of food they can eat in a day.

Conclusion This experiment shows that the simulator is capable of accurately simulating the energy consumption of a fish using an existing model. The implementation of the model was verified by fixing the *Ration_prey* parameter to 0 and 5. The results from these configurations led to the same results as *FB4*. When attempting to replicate *FB4*s results using a more realistic scenario by increasing the *Ration_prey* parameter with the scalar previously computed in the third experiment, it is observed that the results start to deviate. It is hypothesised that this has a similar cause to the previous experiment, where the fish

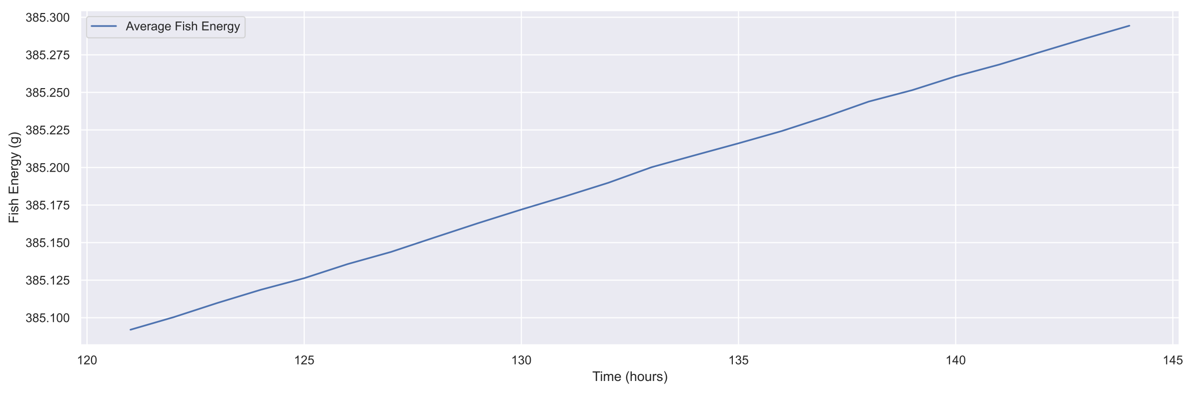

Figure 14: *Weight of a single individual in the school of fish (g). Minor fluctuations are observed in the weight of the fish, which is a result of the simulated movement of the fish.*

are not always able to eat due to the *FeedingRadius* parameter.

Tweaking the parameters until the simulator produces the “correct” results is not a realistic solution. This experiment also shows that the *Magic Scalar* derived from the previous experiment is not a universal solution. As mentioned in the previous experiment, the only real solution is to implement a more realistic feeding behaviour and they would eat at a significantly faster rate.

7 Future Work

FishEcoModeler is a novel approach to simulating fish behaviour. Experiments have already shown that some existing functionality from other simulators can successfully be incorporated into the simulator. There are however many more features that can be added, and many more experiments that can be performed. This section will discuss some of the possible future work.

7.1 Fish Bioenergetics Models

As *FB4* is the current state of the art in the field of fish bioenergetics and served as a major inspiration for the simulator, it would only be logical that more of the models that it supports are implemented. For this thesis, it was decided that only the functions required for the Rainbow Trout would be implemented, but *FB4* has many more available configurations for 75 different species of fish. Good suggestions include the Walleye [29], Burbot [36] and Yellow Perch [29].

A problem with these models is that they are all based on an interval of a single day. These have to be adjusted accordingly, by scaling the final gain in weight of the fish adequately to a single simulator timestep. In experiment 6.5 a minor difference in results between the simulator and *FB4* was observed. This is due to the simulator having a special function to determine the current *Ration_prey* (daily intake in grams) of the fish, which is not present in *FB4*. This function required a minor change to the existing bioenergetics model, a step necessary for all future models that are to be implemented.

The *Ration_prey* parameter could also be replaced when a more realistic feeding behaviour is implemented. This would be a more accurate representation of the fish their feeding behaviour, and would also allow for more accurate results.

7.2 Fish Behaviour

The fish behaviour currently implemented in the simulator is fairly simple and can be improved upon. Huth and Wissel [22] showed that the introduction of a stochastic factor in the behaviour, e.g. drawing from a distribution when computing alignment instead of taking an average like in Equation 13, can lead to more realistic behaviour. A similar approach to simulation is taken by Aoki [1] for their model.

Other existing (terrestrial) simulators such as *Animal Farm*[26], [27] and *EcoTwin*[17] use a state machine to determine the behaviour of the fish. This allows for a distinct split in behaviours, based on the current state of the fish. A similar approach could be taken for the simulator as well.

Another improvement to the simulator would be to introduce obstacle avoidance. The fish currently ignore the characteristics of the terrain and can swim through food sources. If a more complex environment is required, such as a river with rocks, this would be a necessary addition.

Although a basic form of reproduction has been implemented in the simulator, in order to limit scope this was not experimented with or expanded upon. The current implementation is asexual reproduction, where the fish simply split into two identical copies of themselves. The other two simulators outlined earlier feature more complex mating and genetic algorithms. This could be an interesting addition to the simulator, as it opens up the possibility of simulating evolution.

Finally, an alternative way to the fish schooling algorithm could be a predictive network, similar to (new) examples in *EcoTwin* [17].

7.3 Environment

The environment in the simulator is relatively simple right now. It is an infinite space with food sources randomly placed. It is however known that certain fish species use their terrain to their advantage, for example, the Rainbow Trout uses gravel in a river to lay its eggs in and then covers it using by stirring up gravel upstream, effectively using the environment to shield the eggs against potential predators.

As previously mentioned, terrain, as well as water physics in the form of drag and currents would be worthwhile additions to the simulator. This would allow for more complex experiments to be performed, such as the effect of a dam or more rainfall on the fish population or provide a more meaningful value for the *activity* parameter.

In terms of ecosystem simulation, the support for multiple species to create a more complex food web would also be suitable for future research. The consumption pattern of rainbow trout changes throughout its lifetime [20], [42], starting with larvae and plankton, then moving on to crustaceans and smaller fish. This could be simulated by adding more species to the simulator and having the fish change their diet as they age. This would also include the role of new predators, such as birds or larger fish, as of right now the fish can only die of starvation.

7.4 Implementation

Currently with the simulator, the physics and fish decision-making are fixed to the same *RateManager*. In the future, it would be interesting to split these to allow for more accurate physics simulation.

Due to the growing scale of the project, the structure of the code is becoming increasingly important. Currently, the files are all over the place making it more difficult to find the correct file for a specific task. This could be improved by restructuring the project.

Finally, as concluded in the benchmarks in subsection 5.3, some scaling issues arise when the simulator is run with a large school of fish. This is due to the way the simulator allocates memory. This could be improved by using a different memory allocation strategy, such as pre-allocating memory for the fish and reusing it every frame.

8 Conclusion

In this work, a novel 3D simulation framework, *FishEcoModeler*, was proposed that combines the performance of Unity DOTS with existing bioenergetics models. It serves as an alternative approach to the current state of the art *FB4* [15]. After conducting a series of benchmarks and experiments, it is concluded that the framework can accurately model the behaviour of a large school of fish in a fraction of the time that other frameworks do.

8.1 Research Questions

1. With modern technology, is it possible to create a simulation framework that is capable of simulating large schools of fish in complex 3D environments?

As evident from the experiments, the framework is capable of simulating large schools of fish of sizes up to at least 1000 individuals. Simulating larger schools is possible, but as expected it will increase the computational requirement. Although the environment presented in this work is not particularly complex in its geometry, it does have the ability to simulate spatial dependencies of environmental properties such as temperature and light, something which is infeasible with existing tools such as *FB4* and EcoPath. The incorporation of the existing Rainbow Trout bioenergetics also invalidates the old claim that these models are too complex for real-time simulation [31].

2. Can we preserve the accuracy and functionality of existing bioenergetics models whilst incorporating the effects of environmental changes?

As is evident from the experiment done in section 6.5, an existing bioenergetics model has been successfully incorporated into the framework. This model makes use of a temperature parameter, something that can be simulated in the framework. Some minor parameter adjustments have to be made such as the method that is used to calculate the amount of food that is available for the fish to consume which, as shown in the experiment, can be done whilst only slightly affecting the accuracy (2g after 30 days) of the model. This shows that the framework is capable of incorporating existing bioenergetics models and highlights the possibility to implement a more complex temperature gradient.

3. How does the simulation framework compare to existing fish population simulation tools in terms of performance, ease of use, and ecological accuracy, and what unique advantages does it offer to researchers and educators in the field of ecology?

Similarly to *EcoTwin*[46] and *AnimalFarm*[27], the simulator is open for modification, in the sense that the source code is available and that it can be built on a researcher's system granted they have the Unity editor installed. *EcoTwin* also has their website, *ecotwin.se* which contains links to presentations and previous work done on that simulator. *Animal Farm* is only available via GitHub and has two papers that are related to it published [27]. As this is the first piece of work done on my simulator, only this thesis is provided. The source code is available on GitHub[7].

Performance-wise, both existing simulators use the Unity AI package, which means they are *MonoBehaviour-based* architectures. The creators of *EcoTwin* mentioned that performance is “poor” without performing benchmarks. *Animal Farm* seems to run in real-time according to Kiss and Pusztai. Contrary to these simulators, the framework presented in this thesis is not based on the Unity AI package and is therefore not limited by its performance. The proposed framework is capable of simulating one hour of a school of 1000 fish in $1-2$ minutes, depending on the complexity of the energy model.

Although not as extensive or visually pleasing as the other simulators, performance and modularity truly are this simulator’s advantages. Due to the architecture, it is easily extensible and using the provided example Rainbow Trout model, other models can be implemented with relative ease. This architecture also allows researchers to get their results significantly faster, which makes it easier to tinker with the models and experiment with different parameters.

8.2 Reflection

This thesis has been in the works for over two years, mainly due to the complexity of combining multiple research disciplines into a single software solution, but also due to obligations to finish other courses and to work a part-time job.

Changes During this time, the state of the art has changed significantly. The *Animal Farm*[26] and *EcoTwin*[17] simulators were released, both of which are by chance also built in Unity and allow research to experiment with agent-based ecosystem modelling. If these simulators would have been released a couple of months prior to the inception of this thesis, it would have been possible to use some of the work they have done, or perhaps use one of them as a base instead of starting from scratch.

DOTS has also undergone significant changes since it was released. The first version of the simulator was built on experimental version *0.51* and the current version is built on *1.1*. During this time some major breaking changes were made to DOTS, such as a complete rewrite of the transform system, which is used to represent entities in 3D space. Other changes include the introduction of “Bakers” which replace the old GameObject to entities flow that was present in the Unity Editor. The simulator has been updated to use these new features, but this was a time-consuming process.

Experiments The experiments that were conducted in this thesis were not as extensive as they could have been. This was due to the time spent working on the simulator, as it is required before it would be feasible to experiment with.

[7]https://github.com/Trottero/master-thesis-simulator

FB4 played an important role in the experiments as it was used as the “ground truth” for the data. This is a solution to the problem that it is very difficult to get real-world data, as it involves setting up multiple research stations and collecting data over a long period [20].

On a more general note, I am really happy with the results of this thesis and *FishEcoModeler*. I am convinced that it is a great starting point for future research. I am also grateful that I got the opportunity to give Unity DOTS a try in a non-conventional way, as it is a very interesting technology that I will continue to follow closely.

References

[1] I. Aoki, “A simulation study on the schooling mechanism in fish”, *NIPPON SUISAN GAKKAISHI*, vol. 48, no. 8, pp. 1081–1088, 1982. DOI: 10.2331/suisan.48.1081.

[2] D. S. Barrett, M. S. Triantafyllou, D. K. P. Yue, M. A. Grosenbaugh, and M. J. Wolfgang, “Drag reduction in fish-like locomotion”, *Journal of Fluid Mechanics*, vol. 392, pp. 183–212, 1999. DOI: 10.1017/S0022112099005455.

[3] M. Begon, S. M. Sait, and D. J. Thompson, “Predator-prey cycles with period shifts between two-and three-species systems”, *Nature*, vol. 381, no. 6580, pp. 311–315, May 1996. DOI: 10.1038/381311a0.

[4] M. Bergmann and A. Iollo, “Modeling and simulation of fish-like swimming”, *Journal of Computational Physics*, vol. 230, no. 2, pp. 329–348, 2011, ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2010.09.017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999110005115.

[5] M. D. Blank, K. M. Kappenman, K. Plymesser, K. Banner, and J. Cahoon, “Swimming Performance of Rainbow Trout and Westslope Cutthroat Trout in an Open-Channel Flume”, *Journal of Fish and Wildlife Management*, vol. 11, no. 1, pp. 217–225, Oct. 2019, ISSN: 1944-687X. DOI: 10.3996/052019-JFWM-040. eprint: https://meridian.allenpress.com/jfwm/article-pdf/11/1/217/2511513/i1944-687x-11-1-217.pdf. [Online]. Available: https://doi.org/10.3996/052019-JFWM-040.

[6] D. P. Bureau, S. J. Kaushik, and C. Y. Cho, “1 - bioenergetics”, in *Fish Nutrition (Third Edition)*, J. E. Halver and R. W. Hardy, Eds., Third Edition, San Diego: Academic Press, 2003, pp. 1–59, ISBN: 978-0-12-319652-1. DOI: https://doi.org/10.1016/B978-012319652-1/50002-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123196521500021.

[7] J. Carter, *Two visual programming languages for simulation modeling: Stella 5.0 and model-maker 3.0*, 1998.

[8] H. Chalupsky and R. Macgregor, “Stella - a lisp-like language for symbolic programming with delivery in common lisp, c++, and java”, Mar. 2000.

[9] S. R. Chipps and D. H. Wahl, “Bioenergetics modeling in the 21st century: Reviewing new insights and revisiting old constraints”, *Transactions of the American Fisheries Society*, vol. 137, no. 1, pp. 298–313, 2008. DOI: https://doi.org/10.1577/T05-236.1. eprint: https://afspubs.onlinelibrary.wiley.com/doi/pdf/10.1577/T05-236.1. [Online]. Available: https://afspubs.onlinelibrary.wiley.com/doi/abs/10.1577/T05-236.1.

[10] V. Christensen and C. Walters, “Ecopath with ecosim: Methods, capabilities and limitations”, *Ecological Modelling*, vol. 172, pp. 109–139, Mar. 2004. DOI: 10.1016/j.ecolmodel.2003.09.003.

[11] R. Costanza, D. Duplisea, and U. Kautsky, “Ecological modelling on modelling ecological and economic systems with stella”, *Ecological Modelling*, vol. 110, no. 1, pp. 1–4, 1998.

[12] R. Costanza and S. Gottlieb, “Modelling ecological and economic systems with stella: Part ii”, *Ecological Modelling*, vol. 112, no. 2-3, pp. 81–84, 1998.

[13] R. Costanza and A. Voinov, “Modeling ecological and economic systems with stella: Part iii”, *Ecological Modelling*, vol. 143, no. 1, pp. 1–7, 2001, ISSN: 0304-3800. DOI: https://doi.org/10.1016/S0304-3800(01)00358-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304380001003581.

[14] D. L. DeAngelis and W. M. Mooij, “Individual-based modeling of ecological and evolutionary processes”, *Annual Review of Ecology, Evolution, and Systematics*, vol. 36, no. 1, pp. 147–168, 2005. DOI: 10.1146/annurev.ecolsys.36.102003.152644. eprint: https://doi.org/10.1146/annurev.ecolsys.36.102003.152644. [Online]. Available: https://doi.org/10.1146/annurev.ecolsys.36.102003.152644.

[15] D. Deslauriers, S. R. Chipps, J. E. Breck, J. A. Rice, and C. P. Madenjian, “Fish bioenergetics 4.0: An r-based modeling application”, *Fisheries*, vol. 42, no. 11, pp. 586–596, 2017. DOI: https://doi.org/10.1080/03632415.2017.1377558. eprint: https://afspubs.onlinelibrary.wiley.com/doi/pdf/10.1080/03632415.2017.1377558. [Online]. Available: https://afspubs.onlinelibrary.wiley.com/doi/abs/10.1080/03632415.2017.1377558.

[16] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory”, in *MHS’95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43. DOI: 10.1109/MHS.1995.494215.

[17] H. Glimmerfors and V. Skoglund, “Combining reflexes and reinforcement learning in evolving ecosystems for artifical animals”, Ph.D. dissertation, Master’s thesis, Chalmers, 2021.

[18] V. Grimm and S. F. Railsback, *Individual-based Modeling and Ecology*. Princeton: Princeton University Press, 2005, ISBN: 9781400850624. DOI: doi:10.1515/9781400850624. [Online]. Available: https://doi.org/10.1515/9781400850624.

[19] P. C. Hanson, T. B. Johnson, D. E. Schindler, and J. F. Kitchell, “Fish bioenergetics 3.0 for windows”, 1997.

[20] M. Heinichen, M. C. McManus, S. M. Lucey, *et al.*, “Incorporating temperature dependent fish bioenergetics into a narragansett bay food web model”, *Ecological Modelling*, vol. 466, p. 109 911, 2022, ISSN: 0304-3800. DOI: https://doi.org/10.1016/j.ecolmodel.2022.109911. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304380022000357.

[21] S. Hewett, B. Johnson, and U. of Wisconsin Sea Grant Institute, *Fish Bioenergetics Model 2: An Upgrade of A Generalized Bioenergetics Model of Fish Growth for Microcomputers* (Fish Bioenergetics Model 2: An Upgrade of A Generalized Bioenergetics Model of Fish Growth for Microcomputers v. 1). University of Wisconsin Sea Grant Institute, 1992. [Online]. Available: `https://books.google.nl/books?id=ntyoGQAACAAJ`.

[22] A. Huth and C. Wissel, "The simulation of the movement of fish schools", *Journal of Theoretical Biology*, vol. 156, no. 3, pp. 365–385, 1992, ISSN: 0022-5193. DOI: `https://doi.org/10.1016/S0022-5193(05)80681-2`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0022519305806812`.

[23] B. L. Johnson and S. W. Hewett, *A generalized bioenergetics model of fish growth for microcomputers*, Jan. 1987. [Online]. Available: `https://repository.library.noaa.gov/view/noaa/44099`.

[24] J. Kennedy and R. Eberhart, "Particle swarm optimization", in *Proceedings of ICNN'95-international conference on neural networks*, IEEE, vol. 4, 1995, pp. 1942–1948.

[25] D. W. King, D. D. Hodson, and G. L. Peterson, "The role of simulation frameworks in relation to experiments", in *2017 Winter Simulation Conference (WSC)*, 2017, pp. 4153–4174. DOI: `10.1109/WSC.2017.8248123`.

[26] A. Kiss and G. Pusztai, "Animal farm—a complex artificial life 3d framework", *Acta Universitatis Sapientiae, Informatica*, vol. 13, no. 1, pp. 60–85, 2021. DOI: `doi:10.2478/ausi-2021-0004`. [Online]. Available: `https://doi.org/10.2478/ausi-2021-0004`.

[27] A. Kiss and G. Pusztai, "Using the unity game engine to develop a 3d simulated ecological system based on a predator-prey model extended by gene evolution", *Informatics*, vol. 9, no. 1, p. 9, Jan. 2022, ISSN: 2227-9709. DOI: `10.3390/informatics9010009`. [Online]. Available: `http://dx.doi.org/10.3390/informatics9010009`.

[28] J. Kitchell, J. Koonce, J. Magnuson, R. ONeill, H. Shugart, and R. Booth, "Model of fish biomass dynamics", *Transactions of The American Fisheries Society - TRANS AMER FISH SOC*, vol. 103, pp. 786–798, Oct. 1974. DOI: `10.1577/1548-8659(1974)103<786:MOFBD>2.0.CO;2`.

[29] J. Kitchell, D. Stewart, and D. Weininger, "Applications of a bioenergetics model to yellow perch (perca flavescens) and walleye (stizostedion vitreum vitreum)", *Journal of the Fisheries Research Board of Canada*, vol. 34, pp. 1910–1921, Apr. 1977. DOI: `10.1139/f77-258`.

[30] J. N. Kremer and S. W. Nixon, *A coastal marine ecosystem: simulation and analysis*. Springer Science & Business Media, 2012, vol. 24.

[31] T. Laevastu and H. A. Larkins, *Marine fisheries ecosystem*. 1981.

[32] A. J. Lotka, *Elements of physical biology*. Williams & Wilkins, 1925.

[33] C. A. Mack, "Fifty years of moore's law", *IEEE Transactions on semiconductor manufacturing*, vol. 24, no. 2, pp. 202–207, 2011.

[34] R. M. May, “Limit cycles in predator-prey communities”, *Science*, vol. 177, no. 4052, pp. 900–902, 1972, ISSN: 00368075, 10959203. [Online]. Available: http://www.jstor.org/stable/1734806 (visited on 07/24/2023).

[35] W. H. Neill, T. S. Brandes, B. J. Burke, *et al.*, “Ecophys.fish: A simulation model of fish growth in time-varying environmental regimes”, *Reviews in Fisheries Science*, vol. 12, no. 4, pp. 233–288, 2004. DOI: 10.1080/10641260490479818. eprint: https://doi.org/10.1080/10641260490479818. [Online]. Available: https://doi.org/10.1080/10641260490479818.

[36] J.-P. J. Pääkkönen, O. Tikkanen, and J. Karjalainen, “Development and validation of a bioenergetics model for juvenile and adult burbot”, *Journal of Fish Biology*, vol. 63, no. 4, pp. 956–969, 2003. DOI: https://doi.org/10.1046/j.1095-8649.2003.00203.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1046/j.1095-8649.2003.00203.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1046/j.1095-8649.2003.00203.x.

[37] T. Pitcher, “Fish schooling”, *Encyclopedia of ocean sciences: marine biology*, pp. 337–349, 2001.

[38] R. Poli, J. Kennedy, and T. Blackwell, “Particle swarm optimization”, *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[39] T. Pottinger, M. Rand-Weaver, and J. Sumpter, “Overwinter fasting and re-feeding in rainbow trout: Plasma growth hormone and cortisol levels in relation to energy mobilisation”, *Comparative Biochemistry and Physiology Part B: Biochemistry and Molecular Biology*, vol. 136, no. 3, pp. 403–417, 2003, ISSN: 1096-4959. DOI: https://doi.org/10.1016/S1096-4959(03)00212-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1096495903002124.

[40] S. F. Railsback and V. Grimm, *Agent-Based and Individual-Based Modeling: A Practical Introduction*. Princeton University Press, 2011, ISBN: 0691136742. [Online]. Available: http://www.jstor.org/stable/j.ctt7sns7 (visited on 07/24/2023).

[41] S. F. Railsback and K. A. Rose, “Bioenergetics modeling of stream trout growth: Temperature and food consumption effects”, *Transactions of the American Fisheries Society*, vol. 128, no. 2, pp. 241–256, 1999. DOI: https://doi.org/10.1577/1548-8659(1999)128<0241:BMOSTG>2.0.CO;2. [Online]. Available: https://afspubs.onlinelibrary.wiley.com/doi/abs/10.1577/1548-8659%281999%29128%3C0241%3ABMOSTG%3E2.0.CO%3B2.

[42] B. RJ, “Native trout of western north america”, *Am Fish Soc Monogr*, vol. 6, pp. i–xx+, 1992.

[43] R. Schaller, “Moore’s law: Past, present and future”, *IEEE Spectrum*, vol. 34, no. 6, pp. 52–59, 1997. DOI: 10.1109/6.591665.

[44] R. M. Sibly, V. Grimm, B. T. Martin, *et al.*, "Representing the acquisition and use of energy by individuals in agent-based models of animal populations", *Methods in Ecology and Evolution*, vol. 4, no. 2, pp. 151–161, 2013. DOI: https://doi.org/10.1111/2041-210x.12002. eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210x.12002. [Online]. Available: https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210x.12002.

[45] D. J. Stewart, D. Weininger, D. V. Rottiers, and T. A. Edsall, "An energetics model for lake trout, salvelinus namaycush: Application to the lake michigan population", *Canadian Journal of Fisheries and Aquatic Sciences*, vol. 40, no. 6, pp. 681–698, 1983. DOI: 10.1139/f83-091. eprint: https://doi.org/10.1139/f83-091. [Online]. Available: https://doi.org/10.1139/f83-091.

[46] C. Strannegård, N. Engsner, J. Eisfeldt, *et al.*, "Ecosystem models based on artificial intelligence", in *2022 Swedish Artificial Intelligence Society Workshop (SAIS)*, 2022, pp. 1–9. DOI: 10.1109/SAIS55783.2022.9833026.

[47] K. W. Thornton and A. S. Lessem, "A temperature algorithm for modifying biological rates", *Transactions of the American Fisheries Society*, vol. 107, no. 2, pp. 284–287, 1978. DOI: https://doi.org/10.1577/1548-8659(1978)107<284:ATAFMB>2.0.CO;2. [Online]. Available: https://afspubs.onlinelibrary.wiley.com/doi/abs/10.1577/1548-8659%281978%29107%3C284%3AATAFMB%3E2.0.CO%3B2.

[48] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control", in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[49] S. Tuljapurkar, "Population cycles, formal theory of", in *International Encyclopedia of the Social & Behavioral Sciences*, N. J. Smelser and P. B. Baltes, Eds., Oxford: Pergamon, 2001, pp. 11 755–11 758, ISBN: 978-0-08-043076-8. DOI: https://doi.org/10.1016/B0-08-043076-7/02112-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B0080430767021124.

[50] B. Van Poorten and C. Walters, "Estimation of bioenergetics parameters for rainbow trout (oncorhynchus mykiss) using capture-recapture data with comparison to estimates from a laboratory-based model", *The Open Fish Science Journal*, vol. 3, Jul. 2010. DOI: 10.2174/1874401X01003020069.

A Experiment Parameters

All configurations are also available on GitHub[8]. Experiments are run with the following command.

```
./Simulator.exe -batchmode -nographics \
    -config 'relative-path-to-config.json' \
    -logfile './player.log'
```

A.1 Benchmarking: Simple Energy Model

The configuration below serves as a template for the first benchmarking experiment, the `SchoolConfiguration: SwarmSize` parameter is changed for each experiment. Ran 5 times with the following values: 1, 5, 10, 25, 50, 100, 200, 300, 500, 750, 1000

```
{
  "SimulationFrameworkConfiguration": {
    "UpdatesPerSecond": 5,
    "MaxSimulationSpeed": 60,
    "HoursToSimulate": 25
  },
  "BoidsConfiguration": {
    "Speed": 0.84,
    "RotationSpeed": 120,
    "PerceptionRange": 10,
    "SeparationWeight": 1,
    "CohesionWeight": 1,
    "AlignmentWeight": 1,
    "StayInCubeWeight": 1,
    "FoodSourceWeight": 2
  },
  "EnergyConfiguration": {
    "InitialEnergyLevel": 384,
    "ConsumptionRate": 0,
    "FeedingRate": 0
  },
  "ReproductionConfiguration": {
    "MinWeightForReproduction": 500,
    "ReproductionWeightLoss": 200,
    "OffspringWeight": 200,
    "ReproductionEnabled": false
  },
```

[8]https://github.com/Trottero/master-thesis-simulator

```
  "SchoolConfiguration": {
    "CageSize": 10,
    "SwarmSize": 1
  },
  "FoodSourcesConfiguration": {
    "NumberOfFoodSources": 0,
    "EnergyLevel": 50,
    "RegenerationRate": 0.00018928433641975307,
    "MaxEnergyLevel": 50,
    "FeedingRadius": 1
  }
}
```

A.2 Benchmarking: Rainbow Trout Model

The configuration below serves as a template for the second benchmarking experiment, the `SchoolConfiguration: SwarmSize` parameter is changed for each experiment. Ran 5 times with the following values: 1, 5, 10, 25, 50, 100, 200, 300, 500, 750, 1000

```
{
  "SimulationFrameworkConfiguration": {
    "UpdatesPerSecond": 5,
    "MaxSimulationSpeed": 60,
    "HoursToSimulate": 25
  },
  "BoidsConfiguration": {
    "Speed": 0.84,
    "RotationSpeed": 120,
    "PerceptionRange": 10,
    "SeparationWeight": 1,
    "CohesionWeight": 1,
    "AlignmentWeight": 1,
    "StayInCubeWeight": 1,
    "FoodSourceWeight": 2
  },
  "EnergyConfiguration": {
    "InitialEnergyLevel": 384,
    "ConsumptionRate": 0.000006462191358024690 9,
    "FeedingRate": 0.00006826352520768960449155,
    "EnergyEquation": 1
  },
  "ReproductionConfiguration": {
    "MinWeightForReproduction": 500,
    "ReproductionWeightLoss": 200,
    "OffspringWeight": 200,
    "ReproductionEnabled": false
  },
  "SchoolConfiguration": {
    "CageSize": 10,
    "SwarmSize": 1
  },
  "FoodSourcesConfiguration": {
    "NumberOfFoodSources": 1,
    "EnergyLevel": 450,
    "RegenerationRate": 5,
    "MaxEnergyLevel": 800,
    "FeedingRadius": 1
  },
  "RainbowTroutEnergyConfiguration": {
    "RTM": 26,
    "RTO": 22,
    "RA": 0.013,
    "RQ": 2.2,
    "RB": -0.217,
    "ACT": 1.3,
    "SDA": 0.172,
    "Alpha1": 5763,
    "Beta1": 0.986,
    "Cutoff": 4000,
    "Alpha2": 7602,
    "Beta2": 0.5266,
    "Oxycal": 13560,
    "UA": 0.0314,
    "UB": 0.58,
    "UG": -0.299,
    "CQ": 3.5,
    "CA": 0.628,
    "CB": -0.3,
    "CK1": 0.2,
    "CK4": 0.2,
    "CTL": 24.3,
    "CTO": 25,
    "CTM": 22.5,
    "FA": 0.212,
    "FB": -0.222,
    "FG": 0.631
  }
}
```

A.3 Validation of simulator precision

```
{
  "SimulationFrameworkConfiguration": {
    "UpdatesPerSecond": 5,
    "MaxSimulationSpeed": 60
  },
  "BoidsConfiguration": {
    "Speed": 0.84,
    "RotationSpeed": 120,
    "PerceptionRange": 10,
    "SeparationWeight": 1,
    "CohesionWeight": 1,
    "AlignmentWeight": 1,
    "StayInCubeWeight": 1,
    "FoodSourceWeight": 2
  },
  "EnergyConfiguration": {
    "InitialEnergyLevel": 384,
    "ConsumptionRate": 0.000006462191358024 6909,
    "FeedingRate": 0.00003028549382716049 1
  },
  "ReproductionConfiguration": {
    "MinWeightForReproduction": 500,
    "ReproductionWeightLoss": 200,
    "OffspringWeight": 200,
    "ReproductionEnabled": false
  },
  "SchoolConfiguration": {
    "CageSize": 10,
    "SwarmSize": 25
  },
  "FoodSourcesConfiguration": {
    "NumberOfFoodSources": 0,
    "EnergyLevel": 50,
    "RegenerationRate": 0.00018928433641975307,
    "MaxEnergyLevel": 50,
    "FeedingRadius": 1
  }
}
```

A.4 Validation of environmental precision

```
{
  "SimulationFrameworkConfiguration": {
    "UpdatesPerSecond": 5,
    "MaxSimulationSpeed": 120
  },
  "BoidsConfiguration": {
    "Speed": 0.84,
    "RotationSpeed": 120,
    "PerceptionRange": 10,
    "SeparationWeight": 1,
    "CohesionWeight": 1,
    "AlignmentWeight": 1,
    "StayInCubeWeight": 1,
    "FoodSourceWeight": 2
  },
  "EnergyConfiguration": {
    "InitialEnergyLevel": 384,
    "ConsumptionRate": 0.000006462191358024690 9,
    "FeedingRate": 0.0000302854938271604 91
  },
  "ReproductionConfiguration": {
    "MinWeightForReproduction": 500,
    "ReproductionWeightLoss": 200,
    "OffspringWeight": 200,
    "ReproductionEnabled": false
  },
  "SchoolConfiguration": {
    "CageSize": 10,
    "SwarmSize": 0
  },
  "FoodSourcesConfiguration": {
    "NumberOfFoodSources": 1,
    "EnergyLevel": 0,
    "RegenerationRate": 0.0007571373456790122 75,
    "MaxEnergyLevel": 3000,
    "FeedingRadius": 1
  }
}
```

A.5 Basic environment interaction between fish and food sources

This experiment had multiple revisions, only the first revision has been included. The other versions include changes to the *FeedingRadius* parameter, which is discussed in the text. And the *FeedingRate* parameter, also discussed in the text.

```
{
  "SimulationFrameworkConfiguration": {
    "UpdatesPerSecond": 5,
    "MaxSimulationSpeed": 60
  },
  "BoidsConfiguration": {
    "Speed": 0.84,
    "RotationSpeed": 120,
    "PerceptionRange": 10,
    "SeparationWeight": 1,
    "CohesionWeight": 1,
    "AlignmentWeight": 1,
    "StayInCubeWeight": 1,
    "FoodSourceWeight": 2
  },
  "EnergyConfiguration": {
    "InitialEnergyLevel": 384,
    "ConsumptionRate": 0.0000064621913580246909,
    "FeedingRate": 0.000030285493827160491
  },
  "ReproductionConfiguration": {
    "MinWeightForReproduction": 500,
    "ReproductionWeightLoss": 200,
    "OffspringWeight": 200,
    "ReproductionEnabled": false
  },
  "SchoolConfiguration": {
    "CageSize": 10,
    "SwarmSize": 25
  },
  "FoodSourcesConfiguration": {
    "NumberOfFoodSources": 1,
    "EnergyLevel": 450,
    "RegenerationRate": 0.0007571373456790122 75,
    "MaxEnergyLevel": 800,
    "FeedingRadius": 1
  }
}
```

A.6 Rainbow Trout Bioenergetics

This experiment also had multiple revisions tweaking individual parameters of the energy model and the *FeedingRate*. These are discussed in the text, all other parameters are the same as the configuration below.

```
{
  "SimulationFrameworkConfiguration": {
    "UpdatesPerSecond": 5,
    "MaxSimulationSpeed": 60,
    "HoursToSimulate": 721
  },
  "BoidsConfiguration": {
    "Speed": 0.84,
    "RotationSpeed": 120,
    "PerceptionRange": 10,
    "SeparationWeight": 1,
    "CohesionWeight": 1,
    "AlignmentWeight": 1,
    "StayInCubeWeight": 1,
    "FoodSourceWeight": 2
  },
  "EnergyConfiguration": {
    "InitialEnergyLevel": 384,
    "ConsumptionRate": 0.0000064621913580246909,
    "FeedingRate": 0.000057870370370 3703,
    "EnergyEquation": 1
  },
  "ReproductionConfiguration": {
    "MinWeightForReproduction": 500,
    "ReproductionWeightLoss": 200,
    "OffspringWeight": 200,
    "ReproductionEnabled": false
  },
  "SchoolConfiguration": {
```

```json
    "CageSize": 10,
    "SwarmSize": 25
  },
  "FoodSourcesConfiguration": {
    "NumberOfFoodSources": 1,
    "EnergyLevel": 450,
    "RegenerationRate": 0.0007571373456790122 75,
    "MaxEnergyLevel": 800,
    "FeedingRadius": 1
  },
  "RainbowTroutEnergyConfiguration": {
    "RTM": 26,
    "RTO": 22,
    "RA": 0.013,
    "RQ": 2.2,
    "RB": -0.217,
    "ACT": 1.3,
    "SDA": 0.172,
    "Alpha1": 5763,
    "Beta1": 0.986,
    "Cutoff": 4000,
    "Alpha2": 7602,
    "Beta2": 0.5266,
    "Oxycal": 13560,
    "UA": 0.0314,
    "UB": 0.58,
    "UG": -0.299,
    "CQ": 3.5,
    "CA": 0.628,
    "CB": -0.3,
    "CK1": 0.2,
    "CK4": 0.2,
    "CTL": 24.3,
    "CTO": 25,
    "CTM": 22.5,
    "FA": 0.212,
    "FB": -0.222,
    "FG": 0.631
  }
}
```