**Simulation of a lifecycle of ordinary and infected organisms based on the principle of natural selection**

**Introduction**

Have you ever wondered whether it is possible to simulate the development and functioning of our soceity along with other species and forms of life? This task was believed to be practically impossible to accomplish as the behaviour of biological organisms takes into account a lot of different environmental and internal factors, which can not be put into specific written algorithm. However, with the first Artificial Intelligence introduced in 1956, the approach of humanity to this problem has cardinally changed, leading to the appearance of new researches and projects related to this topic [1]. The algorithm was based on a system that learned by itself and produced closest to the best results based on its previous experience [2]. With this, numerous simulations were developed with the topic eventually increasing its efficiency and capabilities [1]. Therefore, I have set out the objective to simulate a lifecycle of two types of organisms: normal and infected, which would interact between each other and reproduce based on their previous experience. The aim is to introduce new version of a functional simulation, which would visually respresent simplified logistics of natural selection and produce the material for the further analysis of this process. The scope of this project is limited by the total amount of two species, 8 digitalised sources of information, and few environmental factors (limited area, constant rates of food generation): therefore, the simulation doesn’t aim to represent the behaviour patterns of specific individuals, but rather of a whole population, isolated and generalised. The developed project introduces an amended approach to the simulation of two types of organisms’ lifecycle using the game engine Unity and four-layered perceptron, which makes it actual in the context of fast-developing AI sphere [13]. With the implementation of the constructed approach, the basics of natural selection and the development of simple population can be demonstrated, which may be particularly useful for amateurs studying the field.

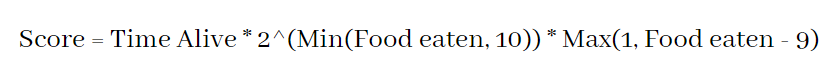
**The initial setup**

As the focus of the project is to develop a simulation of certain types of organisms with various types of behaviour and characteristics, the programming part naturally seemed to be object-oriented. Object-oriented programming (OOP) is a computer programming model, which software design is based on data, or objects, rather than functions and logic [15]. Therefore, I needed to choose an accessible and easily useable programming engine for the project. One of the most advanced engines at the time of this report being written were “Unreal Engine 5“and “Unity” [5]. Unreal Engine 5 is very graphics, sounds and animations oriented and is used for the development of big and tremendous projects [12]. On the other hand, Unity has an easily understandable interface, and its projects use less computing resources. Hence, Unity is generally better for smaller and simpler projects. [11]. The decision was made to use the engine Unity as the simulation needed to be able to run on slower machines and use less memory, which would make it accessible for larger amounts of devices [16]. The programming language chosen for the development of simulation is C# (C-Sharp), developed by Microsoft. It runs on .NET framework (software development framework) and is used to develop web apps, desktop apps, mobile apps, games, etc [10]. The described setup, along with a number of object icons from the web (in a free access) was sufficient to start constructing the application.

**Description of the project**

The primal objective of the project is to simulate a lifecycle of two types of organisms, which would interact with each other within the set boundaries, breed, and evolve. The two types of organisms are normals (peaceful organisms that eat vegetation) and viruses (predators that strive to infect normal organisms).

**Normal.** The normal organisms live and reproduce themselves via the consumption of the simulated food. Each individual acquires an evaluation score throughout their life, which is taken into account by the breeding system at the stage of reproduction. The score is calculated by the formula



, where “Time Alive” is the amount of time the organism has lived, and “Food eaten” is the amount of food units eaten by the organism.

Normal organisms possess 4 senses, which are divided into x and y components to represent various distances on the x and y axes. Two of the senses represent the information about surrounding viruses and other normals, which is calculated by the formula



, where “Value” is the information given to the organism in vector form, and “relative vector” is the distance between the object and organism (in a vector form). Another sense describes the closest food, which is the distance between the food unit and organism in a vector form. The last sense is the centre of mass of viruses, which is calculated by finding the mean distance of viruses to the organism in a vector form.

**Virus.** The virus organisms live and breed by infecting peaceful organisms. Their evaluation score is calculated by the formula



The breeding logic is the same as for normals. The 4 senses that viruses possess are similar to normals, with the exception that viruses consider normals as their food.

When reproducing an organism, the breeding system randomly selects an individual of corresponding type, where the chance of being selected is proportionate to individual’s evaluation score. Two randomly selected individuals exchange their genes to give a birth for their descendant.

Food is randomly generated on the map throughout the simulation, where the rates of generation may be customized before the application starts.

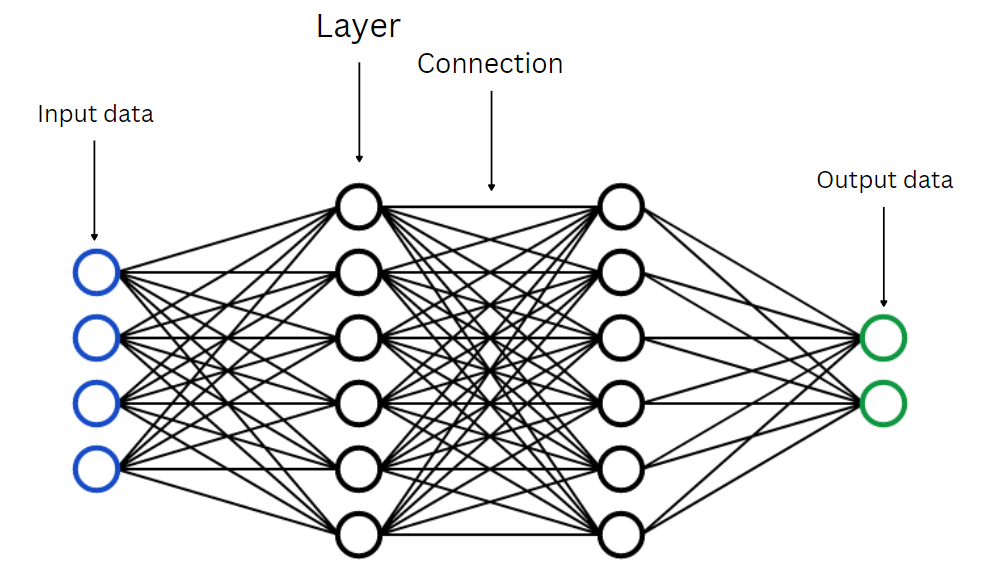
Each individual organism has a brain, which determines the direction of movement every moment (frame).

**Brain and breeding logistics**

*Construction of the brain*

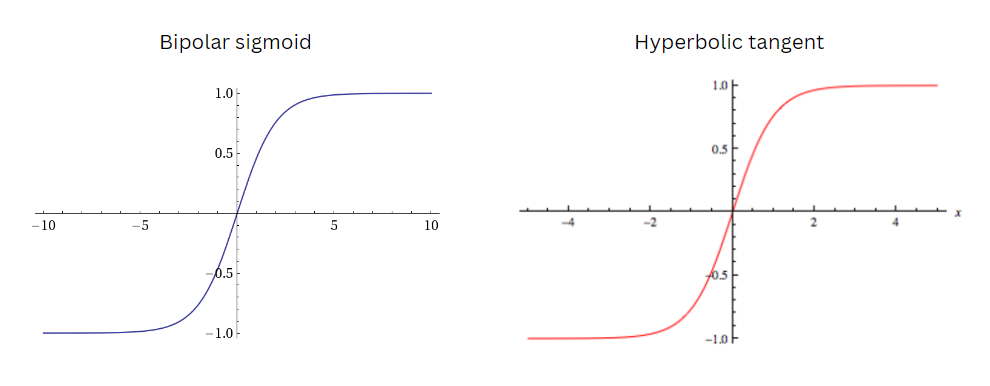
A neural network is a sequence of algorithms that strive to recognize the pattern of given sets of data. Those algorithms can adapt to changing input, which makes them capable of giving a response to previously not encountered sets of data. There are many benefits this functionality can bring as well many ways of teaching the network to emit a right result [2].

To understand the processes happening inside a neural network, one can imagine an object consisting of a few layers. Each of those layers is made of a certain number of cells, where each cell is connected to every other cell in the neighbouring layer. Every connection has a certain value, and the data is transferred from the first layer up to the very last by multiplication with those connections. As the network gets trained, the values of the connections get adjusted, so the last layer displays a desired result [13].



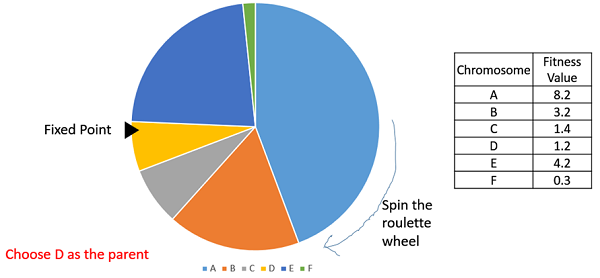
*Example of a neural network*

When progressing throughout the neural network, the data goes through the activation function at each layer. The purpose of the function is to decide whether the neuron (the current cell in the layer) should be activated or not [9]. Depending on the purpose of neural network, the ranges of activation functions may be [-1; 1], [0; 1], etc. For this specific project, bipolar sigmoid activation function was used, alternative to which was hyperbolic tangent [8].



*Bipolar sigmoid and Hyperbolic tangent functions*

One of the possible learning applications for neural network is a genetic algorithm, inspired by Charles Darwin’s theory of natural evolution [7]. This process involves natural selection, where the best-performing individuals are selected for further breeding and reproduction [18]. As the application is most suitable for producing the simulation of natural selection itself, I have chosen this algorithm as a method of organisms’ reproduction. As previously said, when producing a new individual, parents were selected with a chance in proportion to their evaluation function [19]. Then, two parents arbitrary exchanged their behaviour (values of cells’ connections) and produced a child with a mindset similar to their own [18]. It is important to note that a mutation function was implemented, so that each descendant had a four percent chance of acquiring unique behaviour pattern [7].

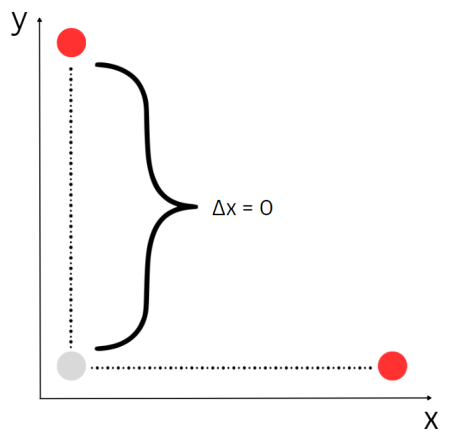


*Method of parent selection*

**The development and decisions made**

With the creation of brain system, I faced a problem of choosing the right activation function. No resource or documentation could suggest the best choice for the project as there are lots of internal factors that are to be considered, so experimental method was the best approach to determine the right decision. A series of experiments were conducted, where I tested the performance with different activation functions and compared the result. For example, the time to reach a certain level of success for organisms (consummate 10 units of food) was measured, and Bipolar sigmoid showed the best result. The populations that used a bipolar sigmoid on average had a lifecycle that, on average, lasted 115 seconds longer than that of hyperbolic tangent (second best alternative). As a result, bipolar sigmoid was chosen as an activation function for this project.

When constructing the data gathering function, I faced another issue. To make the learning process possible, the information about the distance of an individual to other object had to be in inverse, meaning that its value would be higher as the object is closer. However, when the two objects were on same coordinates on y or x axis, their distance on that axis was zero, so the program attempted to inverse 0, which lead to the total disruption. So, I had to invent a new methodology to convey this type of data to organisms, which on the final stage of development was as following: the magnitude of the distance was inversed and multiplicated by the vector of distance. This led to surprisingly successful results, so the creatures were capable of perceiving the data around.



*Visual representation of the described problem*

The method of selecting a parent was also an object of discussion. As with the activation function, there was no specific answer for which of the existing method is the best for parent selection. This is because the best option varies with the nature and specifics of the project, and, thus, has to be determined experimentally. There are 3 generally known methods: fitness proportionate selection, tournament selection and random selection. Fitness proportionate selection chooses organisms with probability proportionate to its evaluation score. Tournament selection randomly chooses a number of individuals, and the one with the highest score becomes a parent. Random selection chooses a parent randomly from the existing population. **Fitness proportionate selection** showed excellent results, where organisms, on average (5 measurements for each method were taken), evolved to the stage of eating 10 units of food in **3 minutes and 21 seconds.** **Tournament selection** was less successful: it took organisms **5 minutes and 6 seconds** to reach the point, where the first organism eats 10 units of food. With **random selection** method, organisms were **not able** to reach the point of measurement. Fitness proportionate selection method was the one with the best results, so it was implemented in the simulation.

**The graphs and analysis**

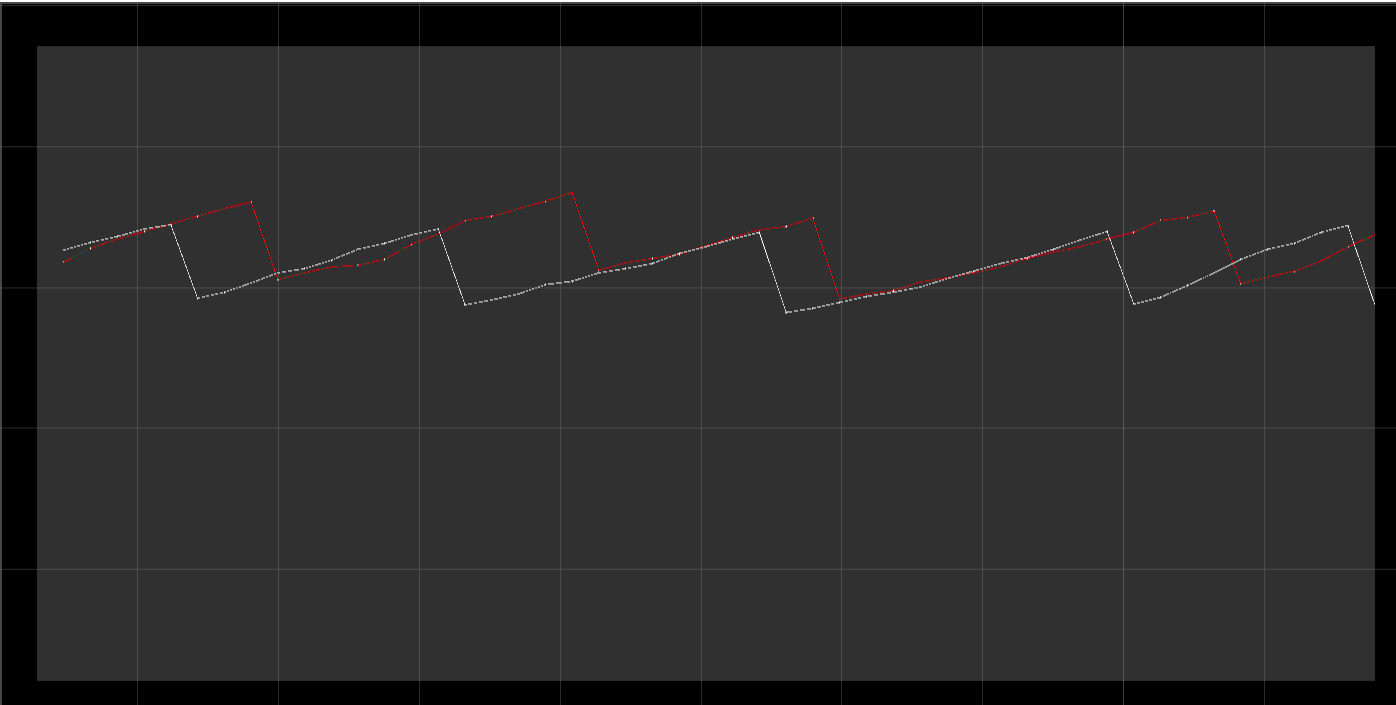
The graph recorded and displayed the information about the population of both normal and virus creatures.

A sequence of trials was made, and with the graph data a number of key population features and behaviours were deducted.

First of all, the number of population of normals was inversely proportional to the number of population of viruses. This meant that when there was a large number of normal creatures, there were only a few viruses and same for the opposite.

Secondly, the amount of each type of species constantly fluctuated, so the population decreased and increased over time. It could be speculated that in some cases the general population consisted mostly of viruses and in others mostly of normals.

Thirdly, when one of the species developed a new effective approach of prolonging its life, it dominated its counterpart for a significant amount of time until the opponent adjusted and evolved as well.



*Graph of virus and normal creatures population*

**Evaluation**

There were three main criteria for the evaluation of the developed simulation. Those were the performance of individuals, data from graphs and feedback from an experienced programmer.

The **performance of individuals** provided the information about the extent to which separate individuals evolved. This included the amount of food separate individuals managed to consume. The results were successful as the organisms of third generation reached the maximum value of one hundred food units eaten. For instance, at the start of the first generation, the maximum count of food units consumed was four. This was one of the arguments that supported the idea that the developed simulation was successful.

The **data from graphs** suggested that the numbers of organisms in population changed in a repeatable manner. This meant that the population produced stable, rather than arbitrary results for analysis, which contributed to the aim of this project and supported the claim that the simulation was successful.

**Feedback from an experienced programmer**, in the first place, suggested that it was a good decision to choose Unity Engine to approach this object-oriented task. Secondly, the programmer (preferred to not share his name) approved the process of choosing the right activation function and parent selection method. On the other hand, he stated that the simulation was overly simplified and couldn’t produce sensible results for the natural selection analysis. This criterion both supported and undermined the idea that the simulation was successful.

**Conclusion**

In conclusion, the simulation of natural selection with two opposing species was developed, which required 1,290 lines of C# code in total. Graphs with the data from simulation were developed and analyzed. As the simulation involved only the total number of two species, limited to 8 sources of information and had only few environmental factors, it was inappropriate to compare it to the development of human or any other population, and thus the determination of desired results was practically impossible. However, according to the three criteria of evaluation, it could be said that the project was successful as a lot of factors supported this claim. As for the future development, the next steps in project is to add a significant amount of environmental factors (like stationary objects, seasonality or weather conditions) and implement new types of creatures (e.g hybrid-consumption, solar energy consumption type). I believe this project to be an important contribution to the AI sphere and will aim to develop it in a broader scale.

*References*

[1] Kelley, K. (2023, March 9). What is Artificial Intelligence: Types, History, and Future. Simplilearn. Accessed 20 March 2023. <https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/what-is-artificial-intelligence>.

[2] Rashid, T. (2016). Make your own neural network (p. 222). CreateSpace Independent Publishing Platform.

[3] Bishop, C. M. (1994). Neural networks and their applications. Review of scientific instruments, 65(6), 1803-1832.

[4] Aggarwal, C. C. (2018). Neural networks and deep learning. Springer, 10(978), 3.

[5] Yengil, M. (2023, February 2). Which Game Engine Is Better? Bootcamp. Accessed 15 March 2023. <https://bootcamp.uxdesign.cc/which-game-engine-is-better-e148d2f75ebc>.

[6] Chen, J. (2022, September 21). What Is a Neural Network? Investopedia. Accessed 28 February 2023. <https://www.investopedia.com/terms/n/neuralnetwork.asp>

[7] Mallawaarachchi, V. (2017, July 8). Introduction to Genetic Algorithms — Including Example Code. Towards Data Science. Accessed 1 March 2023. <https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3>.

[8] Baheti, P. (n.d). Activation Functions in Neural Networks. V7 Labs. Retrieved March 6, 2023, from <https://www.v7labs.com/blog/neural-networks-activation-functions>.

[9] Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. arXiv preprint arXiv:1710.05941.

[10] Microsoft. (2023). C# documentation. <https://learn.microsoft.com/en-us/dotnet/csharp/>.

[11] Unity. (2023). Unity Documentation. <https://docs.unity.com/>.

[12] Unreal Engine. (2023). Unreal Engine 5.1 Documentation. <https://docs.unrealengine.com/>.

[13] Nielsen, M. A. (2015). Neural networks and deep learning (Vol. 25, pp. 15-24). San Francisco, CA, USA: Determination press.

[14] Cox, B. J. (1986). Object oriented programming: an evolutionary approach. Addison-Wesley Longman Publishing Co., Inc..

[15] Gillis, A.S. (2021). Object-oriented programming (OOP). TechTarget. <https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP>.

[16] Eldad, A. (2023, January 30). Unity vs Unreal, What Kind of Game Dev Are You? Incredibuild. <https://www.incredibuild.com/blog/unity-vs-unreal-what-kind-of-game-dev-are-you>.

[17] Whitley, D. (1995). Genetic algorithms and neural networks. Genetic algorithms in engineering and computer science, 3, 191-201.

[18] Mirjalili, S., & Mirjalili, S. (2019). Genetic algorithm. Evolutionary Algorithms and Neural Networks: Theory and Applications, 43-55.

[19] Genetic Algorithms - Parent Selection. (n.d.). TutorialsPoint. Retrieved 15 March, from <https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_parent_selection.htm>.

[20] Flick, J. (2017, November 13). Building a Graph. Catlike Coding. <https://catlikecoding.com/unity/tutorials/basics/building-a-graph/>.