DARWINIO

INTRODUCTION:

When we decided to collaborate on this project, we sat down and brainstormed fascinating and innovative ideas. Some of them were incredibly technical, such as the concept of creating a computer within the Game of Life and then devising a programming language for it. However, that proved to be exceedingly intricate for our skillset. Additionally, we entertained the notion of creating a fluid simulator with accurate physics, but that also proved to be overly ambitious for us.

Ultimately, after careful consideration and evaluation of our capabilities, we settled on creating an evolution simulator. This idea was challenging, but not unmanageable. We were excited to explore the intricacies of the natural world and replicate the processes of evolution through a simulation.

About The Game:

Organisms are driven by their genetic programming to adapt and survive in their environments. It's fascinating to see how they have a desire to reproduce, ensuring the continuation of their species for future generations. They face the challenge of dealing with random variations in temperature and the availability of food, which can be quite unpredictable. Yet, organisms respond to these changes by moving, using a combination of their genetic instructions and their surroundings to guide their behaviors. It's almost as if they have a natural instinct to navigate their world in search of nourishment and optimal living conditions. And just like us, they can even sense and respond to temperature variations, adjusting their movements to find the most suitable climate for their well-being. This beautiful combination between genes and the environment shapes the lives of organisms, allowing them to adapt and thrive in a world that is constantly evolving.

Organism.py:

This code defines a class called `Organism`, which represents an organism. It has attributes like `genome_array` (representing the organism's genome) and `neural_network` (generated from the genome). The `Organism` class has an `__init__` method to initialize its instance. There's a function called `get_random_organism` that creates a random organism. It takes ranges for temperature, trophic level, energy, and reproductive types as inputs. Inside the function, a numpy array of random values within the specified ranges is created, and then an instance of the `Organism` class is created using those values. The `reproduce` function simulates the reproduction between two parent organisms. It takes two parent organisms and a mutation factor as inputs. The function uses the `generate_offspring_genome` function from the `darwinio.genome` module to generate a new genome for the offspring, potentially with mutations

based on the mutation factor. Finally, a new `Organism` instance with the offspring's genome is created and returned. In summary, this code provides a way to represent organisms, generate random organisms, and simulate their reproduction with potential genetic mutations.