EvoSim

A simulation of free agents within a selective and artificial environment

By Lukas Kikuchi

Table of Contents

[Introduction 2](#_Toc322702082)

[Summary 2](#_Toc322702083)

[Background 3](#_Toc322702084)

[Purpose 3](#_Toc322702085)

[Goals 3](#_Toc322702086)

[Delimitations 5](#_Toc322702087)

[Materials and Method 5](#_Toc322702088)

[Implementation 6](#_Toc322702089)

[Results 8](#_Toc322702090)

[User Documentation 8](#_Toc322702091)

[Overview 9](#_Toc322702092)

[Simulatory Components 10](#_Toc322702093)

[How to Make a Simple Simulation 19](#_Toc322702094)

[Conclusion 23](#_Toc322702095)

[References 24](#_Toc322702096)

[Attachements 25](#_Toc322702097)

# Introduction

## Summary

EvoSim is a computer simulation that simulates *artificial life (a-life)*, which is field of study that deals with synthetic systems that behave like biological systems in some way. It deals with life, its processes and its evolution through the use of computers, robotics or biochemistry.[[1]](#footnote-1)

EvoSim simulates these artificial lifeforms through the use of computer simulations. It simulates simple creatures within a simulated environment, within this environment they can survive and compete with eachother and evolve through evolution.

The creatures’s behaviour is controlled by ANNs (Artificial Neural Networks), the attributes and structures of these networks are stored within the creature’sgenome, along with their physical attributes. The creatures have a set amount of behaviours; they can move forward, rotate, eat and mate. They can also see their surroundings.

The creatures evolve through the selection of the environment that they live within, the environment selects the creatures with the most well suited brain (ANN), as those creatures will have the greatest ability to eat (gain energy) , survive longer and mate (spread the genes).

The program is not meant as a definite and as the absolute best simulatory tool for *a-life*; instead it is designed so that it’s easily scalable. For a programmer it is very easy to add new functionality to its featureset. In this way the program has the oppurtunity to have a large scope of use, for the purposes of studying *a-life.*

## Background

For a very long time I’ve been programming and making my own computer applications and I also have a deep interest within artificial intelligence and related subjects. Spontaneous research on different programming techniques led me to stumble upon this (relatively) new model called the *artificial neural network*, which is basically a computational model of a biological brain. This sparked my interest and kept on reading a bit more on ANNs and how they work.

I learned that you don’t program ANNs conventionally using computer code; instead you “train” them, adjusting the brain structure so that they can be used for a chosen purpose. That’s where I got my idea of using evolution to train them, to see if the brain-models could be evolved into a semi-intelligent state, and also to see if evolution would be a practicle and useful way to be able to develop ANNs.

As a way to satiate my interest for artificial intelligence, and my curiosity of how just how intelligent these ANNs could be, I chose to make a computer program that could simulate the evolution of these brain-models.

## Purpose

The purpose of this project is for me to get to work on something that I’m really interested in, and also to create something potentially useful for other people. ANN's are something that are commonly used in many computer programs, that’s why I hope that my program could be used as a tool to train these ANNs.

Another purpose for this project was for me to develop my own application-building skills, as this is a project that involves a whole arrange of different subjects from biology to graphical design (as all programs need some sort of graphical desgin) to math (math played a big part in the ANNs) to programming, this project serves as a way for me to getting used to making big and complex computer programs.

## Goals

EvoSim is first and foremost a computer program that simulates with the life, the processes and the evolution of *free agents*[[2]](#footnote-2) within a selective artificial environment. In the state-of-development (as a program can almost never be said to be “complete”, as there’s always room for improvement) that I present my program, it simulates extremely simplified creatures within an extremely simplified environment, that nevertheless is capable of producing signs of evolution within the simulated creatures.

Before explaining the goals of this program, I will have to explain a core-component of the program that is closely related to the purpose, *artificial neural networks.*

Artificial neural networks or ANNs, are mathematical and computational models that are inspired by biological neural networks, or in other words: brains. In most cases an ANN is a tool for programmers to solve computational problems that are too hard or too complicated to solve with regular programming, usually they are used to interpret information that is organized in a non-structured way (like images), using a heuristic approach. [[3]](#footnote-3)

Just like biological neural networks, ANNs are composed of neurons that transmit information along a multitude of synapses. Simply put, an ANN is an over-simplified version of how a brain works. Just like our brains get information from the outside world and also outputs a response through our limbs, ANNs can recive input and also output a response. The *“*programming” of a particular ANN lies in the way it is structured, it lies within the way the neurons are connected to eachother.

The reason this kind of programming is useful is because it’s good way to solve computational problems that are very abstract and graphical, something that regular programming has a hard time dealing with. If you, for example, would want to make a program that can convert handwritten text into computerized text it would normally be a very hard thing for a programmer to code. So instead of normal programming, you train a “brain” (the ANN) into automatically converting the text.

The simulated creatures within my program are controlled with these ANNs; information about the outside world goes into their “brains” and comes out as instructions on how to act. What is *evolving* within my program is solely the ANN structure and neuron makeup within the creatures.

With this brief explanation of the core-component within my program, I can do delve into the goals of my simulation:

* To create a tool with which you can create and train artificial neural networks, designed for *artificial-life*.
* To create a scalable program, with a featureset that is easily expandable.
* To create a basic selective artificial environment within the program, that is capable of evolving the creatures from an unintelligent to a relatively intelligent level.

The way you develop and create ANNs are through different “training”-methods, there is a wide array of techniques on how to be able to do this, my particular approach is through evolution. One of the purposes of my program is to provide a useful tool to be able to train and create ANNs for *artificial life*.

Early on I realized how enormous of a project this really is, and to create a simulatory tool to accommodate all purposes would be imbossible to create with the time I had. That’s why I added a goal of making the program easily scalable and expandable, so that programmers who would want to use it could customize it in their own way to fit their own needs.

In addition to that I also want to create a basic selective artificial environment for the simulation, to showcase its usage and to give an example on how to use it.

## Delimitations

If I were to make a simulation that would allow extreme complexity within the selective environments and within the *artifical-life*, it would take several years to complete. So instead of trying to do that I’ve chosen to focus on making an easily scalable and expandable program, which can serve as a platform for other programmers to develop for their own purposes.

I used extremely simplified models of real life when I created the simulations. There is no such thing as gravity, physical laws or chemical reactions. The environment is a flat 2D space, covered with food for the creatures to eat.

The creatures themselves are also very simple, their only physical attributes are: age, skincolor and energylevel. Besides those attributes they also have their brain which plays the largest role in their survival and their evolution. In real-life it is both the brain and the physical body that evolves, but as the goal of this project was never to create a realistic simulation of life, it doesn’t matter. The goal of the project is to allow the evolution of these brains (ANNs), from an unintelligent and unordered state into an ordered and intelligent state. The basic preconditions for evolution is a selective environment and genetic variation[[4]](#footnote-4), the environment within my program selects creatures that are the most apt for collecting food to gather energy and their genetic variety lies in their randomly generated genome and ANNs.

## Materials and Method

To create the program I used *Microsoft Visual Studio 2010 Ultimate Edition*, which is a development tool for programming. The reason I chose Visual Studio was because it provides excellent debugging utlities, and tools for analyzing ones code. As I knew that the amount of code within my program was going to grow to a large amount, I found it necessary to write the code within a development suite that provides tools that can analyze the code’s managebality (as a program grows bigger, the code gets less manageable, unless you constantly try to prevent it). Another reason that I chose Visual Studio is because I’m very used to using it.

The programming language that I used was C#. The reason I chose this language is because I’m first of all very used to it, and also because it provides access to Microsoft’s .NET library[[5]](#footnote-5) which is very useful when creating an application.

I read a lot of books on the subjects of evolution, ANNs and genetic mutation (I didn’t read whole books, but I specific parts for reference) to learn on how to code the different parts of the program.

## Implementation

Creating a computer program is a big endeavour for a lone person, to succeed you needs to have a preexisting knowledge of computer programming and program architecture (the way you structure the program) combined with additional knowledge gained through extensive research and planning before even starting to write any code.

What did I have to do to get this project done? I couldn’t just dive into coding immedietly, first I had to *know* how to make the simulation, then I had to *plan* on how to make it and then after that I could actually go about doing it.

I had to do a lot of research on how to be able to implement evolution within the simulation, learning what the basic prerequisits are for evolution to occurr. After that I did a lot of research on how to create ANNs, what kind of ANNs I should use and also how to be able encode them into a genetic format.

It was very easy finding books on the subjects, just a quick google search got me many tons of titles to look through, but unfortunetly there weren’t that many of those books that I could get access to as they were only avalible in the USA, I could still find lots of books within Sweden but they were not of the same quality as those if found online. Luckily there were free online e-books that I could find, and also various internet resources.

After I had assembled enough research on the subject I could start planning the program, deciding what features is should have within it, how it will look, and how the code will be structured internally. In any program there’s a graphical component (that paints the user interface upon the computer screen) and also an internal calculating component, so I started with planning these individual components. But I didn’t have the foresight or knowledge to completely know what kind of features my program should have, so I decided to make a preliminary plan on how my program should be constructed, and then whenever I’ve completed the program up to that stage I write another plan that adds new features and continue on from there.

I also made up a time-plan for me to follow, but it is hard to set concrete weekly or monthly goals as in the programming process new problems and new necessary components arise constantly that you have to tend to, so it would’ve been impossible for me to exactly say when a particular part of the program should be done. That’s why instead of making a detailed time-plan I decided to instead just make sure that I wrote atleast 200 lines of code per week, as I made an estimate that the project would be around 10’000 lines of code in total (in the end it turned out to be 12’566 lines of code.).

I had assembled a lot of research to support my coding, but as I was on a time limit I had started coding before I could really get my head into a lot of the information within the books and the texts. I followed my plan of how the program would look like and started coding, whenever I would hit a point were my knowledge was lacking I would read up on the subject to be able to tackle that particular part of the program then and there.

To say that my preliminary research was enough to support my whole development and construction of the program would be a lie, many times I had to take a break from the programming and seek new material to be able to proceed, because I found things that were missing from the code that were essential for the programs function. As I am a lone developer it can’t be expected of me to be able to be knowledgable about programming, evolutionary and neurological biology, mathematics and graphical design at the same time.

The texts and the books that I read we’re academic in nature so it was often very hard to understand, and also very easy to misundestand. When I coded the most complex part of the program, the ANN, I made many errors in the code that made the ANN non-functional in the beginning, because I had misunderstood a part of the research material I had. In the end I had to reread the material several times to fully understand it, and only then could I write the correct code.

As there are different kinds of ANN models to choose from, I had to decide on which kind I wanted to use. In the beginning I had a *Spiking Recurrent Neural Network*, the one that closely resembles the brain, as it simulates the actual electric impulses within the brain in real-time. But it turned out that performance-wise this wasn’t a very wise choice, because this ANN used up a lot of the processing power from the computer. So I instead opted for another model, the *Recurrent Multilayer Perceptron* that used an approximative mean for the synapse-activity between the neurons, allowing much faster computing performance.

After the program was done I proceeded with writing a User-Documentation file in Word, which is basically a tutorial on how to use the program.

I pretty much stuck to the projectplan’s description of my work-method. The only difference is that I didn’t foresee that I was going to have to constantly find new texts on the subjects, and that the research portion of the project would extend to the whole span of the project time-lapse.

# Results

I succeded in completing the program and it is fully functional, and I’m very satisfied with what I’ve accomplished. In the end I could accomplish the goals I had; the program is a tool for creating ANNs, it’s easily scalable and it’s got a ready-made artifical selective environment to showcase from the get-go when you start up the program.

In the beginning stages of the project I was unsure of how far I could manage to take the simulation, I knew that I had to atleast get to the point where I could simulate simple artificial creatures, but I had my hopes up to go even farther than that, I wanted to add abiotic elements to the simulation for example. Unfortunetly my computer is very slow, and could barely manage running the current stage of the program, so I couldn’t go any farther. But as I still managed to reach the project goals I’m still satisfied.

With any computer program there are many bugs, especially if you have to rush and get things done quickly. Many parts of the code could be rewritten and be done better to get a faster performance, as threre are some unelegant solutions in the program (I’m not going to go into detail as it’s all very technical). I’ve eliminated most of the bugs, but performance-wise I could’ve done better.

The project was nevertheless a success and in the end I got what I set out to do, and this is definetly something I’m going to keep working on and I’ll also try to get it out there to other programmers so that they can take part in the code.

On the next page you can find my user-documentation of my program, it’s a bit technical in terminonology and it requires some previous knowledge of ANNs.

## User Documentation

### Overview

Here I’m going to go through the basic-parts of the application, so that you get an understanding on as to how the program works.

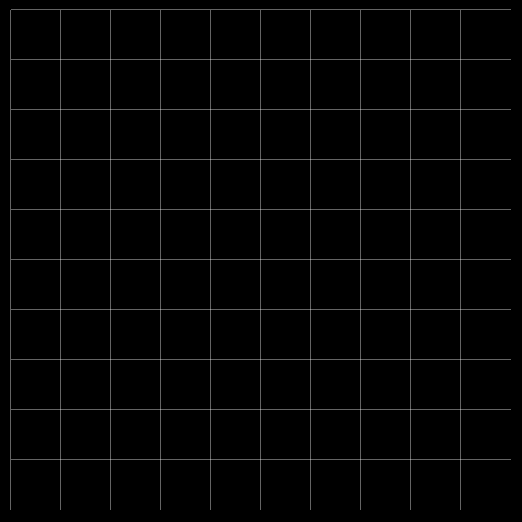
**Simulation**

This is what the application will run. You create a simulation, you run it, you save it, you close it, and then load it up again and continue running it.

A simulation contains a…

**World**

The world is the thing that the artificial creatures live within. It looks like this:



It’s not much too look at…

It’s made up of a grid, with a specified cell-size. The cells within the grid are called **regions**. You usually don’t need to care much about the regions, as they don’t affect the simulation in any way. The size of the regions can affect performance a bit though.

The world is inhabited by…

**Entities**

Not only creatures!

All objects that exist on the map are referred to as *entities*. The artificial creatures are entities, and also the foods that the creatures eat are entities.

**Creatures**

They are the artificial life-forms that live within the world. They are made up of two different things:

* Genome
* ANN

The *Genome* decides the physical parameters of the creature. The *ANN* is the brain of the creature. Besides these two things there are also different Creature-types, the most basic Creature-type is *Creature*.

**Templates**

Whenever you want to add a creature to the world, you have to do it through the use of *templates*. More specifically, you need to use 3 different templates:

* Creature-template
* Genome-template
* ANN-template

A creature, a genome or an ANN has loads of different parameters and options that can be adjusted. Templates are stored configurations of the parameters, which allow quick mass-production of creatures with the same configuration.

There are also world-templates, and entity-templates, we’ll go through these in more detail later.

### The User-Interface Components

The user nterface of the simulation is not overly complicated; it has three main elements, as you see in the picture.

1. This is the menubar, where most of the simulation management occurs.

**File** is where you save or open simulations.

**Edit**

**Simulation** allows you to change simulation-settings, that mostly effects how information is presented to you rather than affecting the actual simulation.

**World** is where you edit or manage the world.

**Entities** allow you to edit and add entities or creatures to the world.

**Templates** allow you to manage all of the different templates.

**Tools** hold a bunch of analytical tools.

**Help** is pretty self-explanatory isn’t it?

1. Here’s the world map, the green circles are food-entities.
2. The sidebar is where quick-information about a selected entity is displayed.

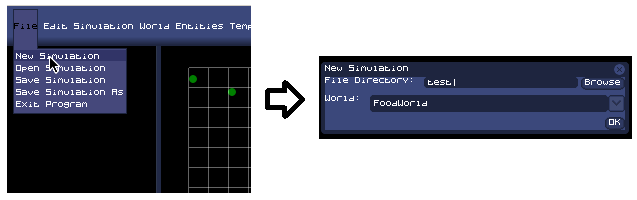
### Simulatory Components in Detail

#### Simulation

To begin simulating these lifeforms you need to create a *simulation*. A simulation is what contains all the simulation data that you run, the simulation is what you save and load on your harddrive.

A simulation doesn’t have that many attributes: the *world* that it simulates, the name of the file, and also drawing options for all the entities and creatures.

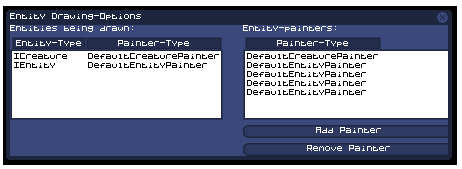
To create a new simulation you just go into File->New Simulation.



##### Entity Drawing-Options

You can choose between different ways to draw your entities, you do that by assigning an *entity-painter* to a specific entity type. Different entity-painter draws in different ways, and many entity-painters can only be assigned to a specific entity.

If you go to Simulation->Entity Draw-Options you get to the menu where you can manage all the entity-painter settings.



Here you can add or remove painters. The left panel shows which entity-types are being drawn by which entity-painters, the right panel shows which entity-painters are active at the moment.

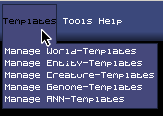
To assign an entity-painter to an entity, you must click the add painter button, and you get to this form:



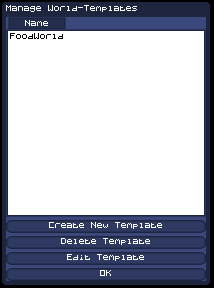
Here you choose what entity-painter-type to use, and then you select which entities it is supposed to draw, and then you just click *ok* to add it!

#### Templates

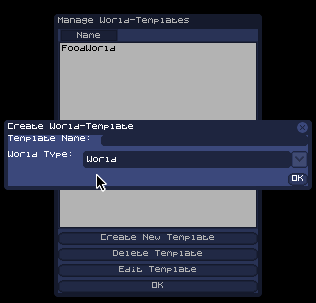
Everything within the simulation is created through templates. What are templates? Templates are ready-made configurations of different parts of the simulation, like; worlds, entities or ANNs.



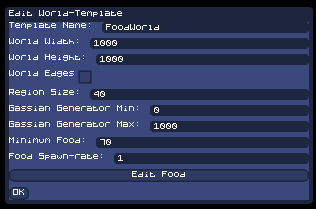
If we click on *Manage World-Templates* we get this:



Here you see that there is one preexisting template already there, you can create a new one by clicking on *Create New Template*.



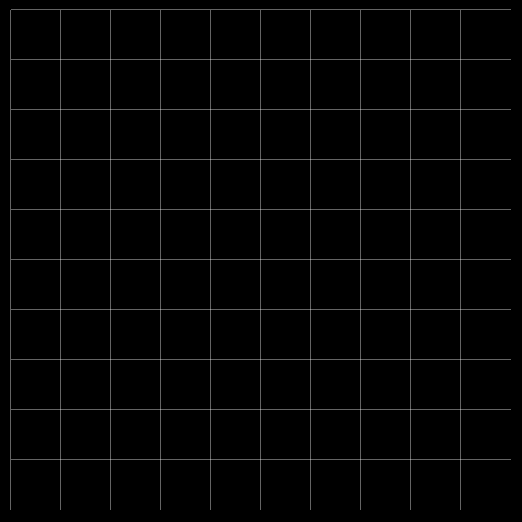
Before getting to the part where you can configure the world-template’s parameters, you first have to decide what name the template should have and what kind of world-type it should be. After your done you can click *ok*, and you get to the part where you edit the paramters.



When you’re done click *ok* and you’ve got yourself a new template!

#### World

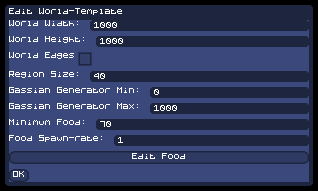
The world is the artificial selective environment that the creatures live within; it’s a 2D surface that is composed of a gridnet. The cells within the grid are called **regions**, and when you create a new world you get decide the size of these regions (the regionsize doesn’t affect the simulation at all, except performance-wise).



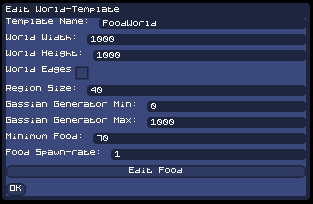
Whenever you create a new simulation, you have to choose what kind of world you want to simulate. There are different world-types, that all work differently. The two default world-types are:

* *World*, is just a 2D landscape that can hold different entities.
* *FoodWorld*, automatically spawns food onto the map.

If you go to World->Edit World, you can change the currently simulated world’s properties. This particular window is for a FoodWorld.



If you edit a world-template you get this window:



This is exactly the same window.

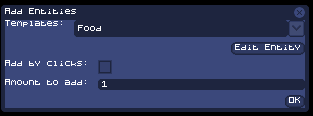
#### Entities and Creatures

Entities is the common name for everything that exists within a world, different entities have different properties and they all serve completely different roles within the simulation.



Here you can see what you can do with entites; you can add them, edit their properties, delete them or choose to delete all of them. Handling Creatures work much the same way as handling entities.

If you want to add an entity you get this window:



You have to choose between different entity-templates, and you can also customize the template specifically by clicking on *Edit Entity*.

If you check *Add by Clicks*, the entities are spawned whenever you click with the left mouse button, to turn that off you click the right mouse button. If you let it go unchecked you instead have to specify the amount of entities should be randomly spawned across the map.

To delete an entity you first have to select which entity to delete by clicking on it in the map and then clicking Entities->Delete Entity.

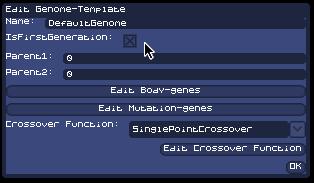


Editing an entity works much the same way, you just click Entities->Edit Entity.

#### Genome

For evolution to occur, creatures need to have a genome that they can pass on through generations. Luckily, this simulation isn’t lacking in that department!

When you create or edit a genome-template you get this window:

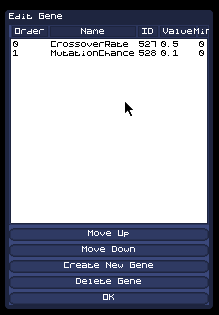


This is also the window get whenever you try to edit a genome, if you click *Edit Body-genes* you can edit the color of the creature.



Here you see three different genes, one for the color red, one for the color blue, one for the color green. You can edit the genes by double-clicking them. You can also delete the genes, if you don’t want the creatures to have them at all.

If you click *Edit Mutation-Genes* you get a similar window:

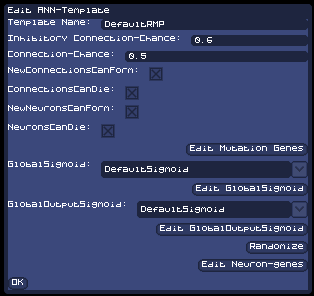


These two genes are parameters that figure into the mating processes of the creatures, *CrossoverRate* decides the rate of gene-crossover between the two creature’s chromosomes; *MutationChance* decides the rate of gene-mutation.

The genome actually only deals with the physical genes of the creature, for the ANN there is a separate ANN-chromosome.

#### ANN

The ANN within the creature is stored within an ANN-chromosome, encoded in genes. If you edit an ANN-tempalte you get a window like this:



*Inhibitory Connection-Chance*: The chance of a random connection being inhibitory.

*Connection-Chance:* The chance of a connection occuring between two neurons.

*NewConnectionsCanForm*: If checked, neurons can mutate and form new connections.

*ConnectionCanDie*: If checked, neurons can mutate and kill connections.

*NeuronsCanDie:* If checked, neurons can mutate and die.

Just like the genome, the ANN-chromosome has Mutation-Genes.

The sigmoids of the ANN can be configured and changed between different types; you also have two different sigmoids, one for output-neurons only and one for the rest of the neurons.

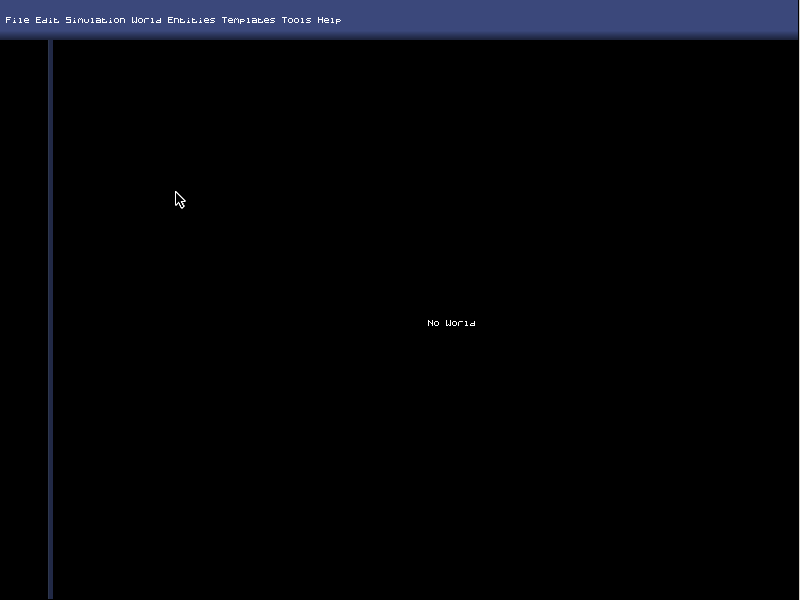
If you click *Randomize*, you get a random brain generated with the parameters you choose.

If you click *Edit Neuron-Genes*, you get to edit all of the genes within the ANN manually.

### How to Make a Simple Simulation

Here we are going to go through the process of making a simple simulation, using the default templates that initially come with the application.

Here is your initial screen when you start the program, notice the *“No World”* notation in the world-screen, we want to change that to a view of a simulated world.



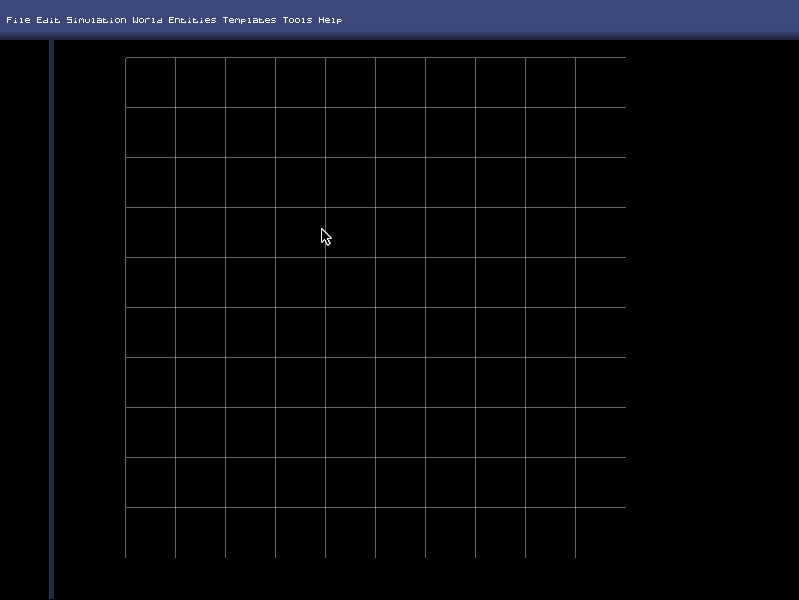
To create a new simulation, click *File->New Simulation*. You’ll get this dialog-box:



In *File Directory* you are supposed to write the file-name of the simulation will be saved as. You can click the *Browse* button to browse to a specific directory, or just write the directory and file-name manually. If you do not start the name with *“C:/”*, the root directory will automatically be the program-directory.

Below *File Directory* there is the *World* combo-box. Here you choose between different World-templates that exist. Choose one and then click ok. If you haven’t created any World-templates you can use any of the pre-existing World-templates that come with the program.

After you’ve clicked ok you have basically created an empty world:



Now we’re going to add some entities, we’ll start by adding some Food.

Go to Entities->Add Entity, you’ll get this dialog-box.



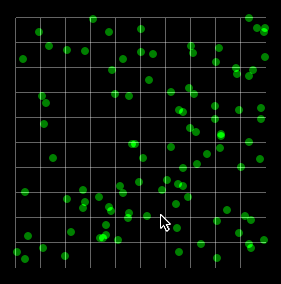
Select what entity-template you want to use, in the *Templates* combo-box. Because we want to create food, you have to choose a template which is of a food type. You can either use the pre-existing template called *food* or use one of your own.

If you want to change the parameters of the entity you want to add, click the *Edit Entity* button, you’ll get a dialog-box in which you can manually change all the parameters of the template. All the changes you make to the template are temporary changes that only apply to the entity you’re currently adding.

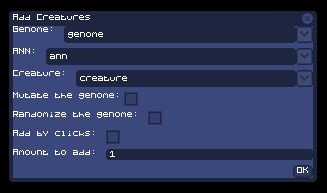
The *Add by Clicks* decides whether you’re going add the entities using “clicks” (clicking on the world-map to add creatures) or by randomly placing a set amount of the entity across the map. If you choose to add using clicking, you don’t need to worry about the *Amount to Add* integer-field. **Note: You exit add-by-click-mode by right-clicking.**

The *Amount to Add* integer-field decides how many entities you want to add to the world-map, after you press Ok, that many entities will be distributed randomly across the map.

Press Ok and the map will be filled with food!



Now we want to add creatures into the mix, this is a bit more complicated (just a bit though!). Click *Entities->Add Creatures*, you’ll get this dialog-box:



This dialog is pretty similar to the *Add Entity* dialog, but with a few added parameters. Because a creature consists of two main things: The Genome and the brain (ANN) You need to decide on which templates you want to use for the different parts of the creature. After that you need to decide on a Creature-type template.

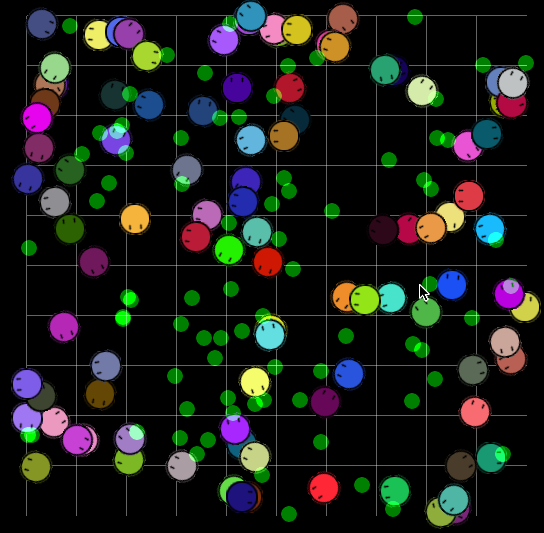
As mentioned earlier, use the pre-existing template that comes with the application or use one of your own templates.

If the *Mutate the genome* checkbox is checked, the *genome* and the *ANN* will be mutated before being added. If you choose to add 100 new creatures, each of these creatures will be mutated (differently) before being added.

If the *Randomize the genome* checkbox is checked, the *genome* and the *ANN* will be totally randomized before being added. Instead of just slightly mutating the genome of the creature it will instead be completely randomized. If you want to create a world where you want an initial gene-variation, you should check this checkbox.

The *Add by clicks* and the *Amount to add* parameters works the same way as when adding an entity.

After you’ve added some creatures you’ve got yourself a full-fledged simulation!



Please note though that in order to actually create a simulation where all the creatures don’t die after 5 minutes you have to do some serious calibration. That is why it is recommended to use the pre-existing and pre-calibrated templates.

# Conclusion

As according the *skolverkets* course-plan, I have for this project used my own imagination and ingenuity, combining several scientific areas, to formulate a specific task for me to accomplish. During the course of the project I have deepend my knowledge within subject’s scientific nature (foremost within evolutionary biology, a bit of neuroscience and also of course ANNs) and I’ve become better at building applications.

I’ve developed my ability to plan, structure and take responsibility for a larger project, one that have given me experience in working in a project-format. My time-plan turned out to be realistic as I got the project done in the end; it was easy to follow as I only needed to make sure I got a certain amount of work done every week, and this worked very well for me.

My project-plan went through many revisions and the initial goals and project-purpose were very different from those that I have now. As the project went on I felt that my vision of my program became clearer and I got more goal-oriented, with the help of the supervision and guidance of my instructor. I felt that the initial preparations and the planning phase were deciding factors in the project’s success, the material and research gathering helped me be prepared and allowed me to take a good first-step when starting to build the program.

During the work-process I have written a logbook describing the whole process of making my program, and I’ve also hade several supervisional talks with my instructor to keep him updated. I feel that now in the end when I’ve finally produced a product, I’ve managed to substantialize my idea of creating a tool for developing Artificial Neural Nets through the help of evolution, and I’ve been able to document its use and function in a User-Documentation file which allows other people (mostly programmers) to take part in the fruits of my project.

The way that I’ve presented my project and my work-process is through this projectreport, through a presentation that I held to my classmates and also through my User-Documentation that I’ve included in this report.

All-in-all the project went really well, what has worked the best is the way I could manage to have a steady amount of work done each week. I’m also very happy with the way that I could use research on a wide arrange of subjects and implement them into my program, whenever I hit an obstace or whenever I couldn’t continue because my code was wrongly structured I went and searched for new material that could help me understand the scientific models better, so that I could in turn implement them better.

What I could have done different is to first of all get a better and faster computer, one of the obstacles when creating the program was the performance, in the end I couldn’t extend the features of my simulation because my computer couldn’t handle it.

I got the project done faster than I initially expected, so I could’ve actually taken more time writing the code, which would’ve allowed me to eliminate bugs better and also to get the code to be more structured and maintainable.

The supervision that I had was useful to me because it allowed me to be more clear about how my project-purpose and my project-goals should be formulated, but naturally I didn’t get much help with my work-methods as my instructor isn’t a programmer himself.

What I set out to do with my project is to make a tool for creating and training ANNs, that does this using evolution. I wasn’t sure if this was a doable goal in the beginning, but now it seems to have worked, so now the question is just how far it would be possible to take my simulation.

The initial artificial selective environment that I set up within my program is very basic and isn’t at all similar to the real world, how would a more complex environment affect the creatures? There are many things that I could add; abiotic influences, the ability to kill one another, a day and night cycle etc etc… Using evolution the creatures would hopefully embrace changes to my code and use them to their own advantage. If I would add the ability to kill, it might incite a tribal behaviour among the creatures, as creatures might choose to kill other creatures of different color (as those with different color problably doesn’t share a common ancestry) in order to ensure their own gene’s prevalance among the global gene pool.

Artifical Neural Networks and evolution are exciting scientific fields that I have thoroughly enjoyed working with, and also will continue enjoy working with.

# References

<http://www.cs.stir.ac.uk/>, "Introduction to Neural Networks" in *The University of Stirling*. Source location: Prof. Leslie Smith. <http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html> Accessed: April 18, 2012

<http://www-genvagar.slu.se/> “Evolution” in *Genvägar*. Source location: <http://www-genvagar.slu.se/lartext/larar5.htm>. Accessed: April 19, 2012

<http://www.dictionary.com/>, "artificial life," in *The Free On-line Dictionary of Computing*. Source location: Denis Howe. [http://dictionary.reference.com/browse/artificial life](http://dictionary.reference.com/browse/artificial%20life). Available:[http://dictionary.reference.com](http://dictionary.reference.com/). Accessed: April 18, 2012.

# Attachements

## Books

“Principles of neurology”. Publisher: McGraw-Hill, 1997. Authors: Raymond D. Adams, Maurice Victor, Allan H. Ropper

“Artificial intelligence on the Commodore 64: make your micro think”. Publisher: Sunshine Books (1984). Authors: Keith & Steven Brain

“Fuzzy neural network theory and application”. Publisher: World Scientific (2004). Authors: Puyin Liu, Hongxing Li

“A Brief Introduction to Neural Networks” Publisher: No publisher (2007). Author: David Kriesel Source Location: <http://www.dkriesel.com/>

“Origin of species” Publisher: Collectors Library (2004). Author: Charles Darwin Source Location: <http://darwin-online.org.uk/>

## Webpages

"Introduction to Neural Networks" in *The University of Stirling (*webpage: <http://www.cs.stir.ac.uk/>). Author: Prof. Leslie Smith. Source location: <http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html>

“Evolution” in *Genvägar* (webpage: <http://www-genvagar.slu.se/>). Source location: <http://www-genvagar.slu.se/lartext/larar5.htm>

“Neural Nets” in *AI Junkie* (webpage: <http://www.ai-junkie.com/>) Source location: <http://www.ai-junkie.com/ann/evolved/nnt1.html>

“How the brain works” in *How Stuff Works* (webpage: <http://www.howstuffworks.com>) Authors: Craig Freudenrich, Ph.D. and Robynne Boyd. Source location: <http://science.howstuffworks.com/environmental/life/human-biology/brain.htm>

“How the evolution works” in *How Stuff Works* (webpage: <http://www.howstuffworks.com>) Authors: Marshall Brain. Source location: <http://science.howstuffworks.com/environmental/life/evolution/evolution.htm>

1. Dictionary.com, "artificial life," in *The Free On-line Dictionary of Computing*. Source location: Denis Howe. [http://dictionary.reference.com/browse/artificial life](http://dictionary.reference.com/browse/artificial%20life). Available:[http://dictionary.reference.com](http://dictionary.reference.com/). Accessed: April 18, 2012. [↑](#footnote-ref-1)
2. Agent is a term that describes an entity that can *act*, that can do actions, and in this case think as well. *Free agent* is not a scientific term, with *free* I mean that there’s a randomness that takes part in the decisionmaking of the agent. [↑](#footnote-ref-2)
3. <http://www.cs.stir.ac.uk/>, "Introduction to Neural Networks" in *The University of Stirling*. Source location: Prof. Leslie Smith. <http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html>. Accessed: April 18, 2012 [↑](#footnote-ref-3)
4. <http://www-genvagar.slu.se/> “Evolution” in *Genvägar*. Source location: <http://www-genvagar.slu.se/lartext/larar5.htm>. Accessed: April 19, 2012 [↑](#footnote-ref-4)
5. *Library*, has code that is ready-made for specific purposes. [↑](#footnote-ref-5)