ALCS REPORT 4 - EVOLUTIONARY DESIGN

In this file, I will present my take on Framsticks report based on my adventure with it. I assume that you have basic knowledge of the program so if you have some trouble understanding how something works watch the tutorials on Framsticks webpage.

1. Specify the evolutionary goal

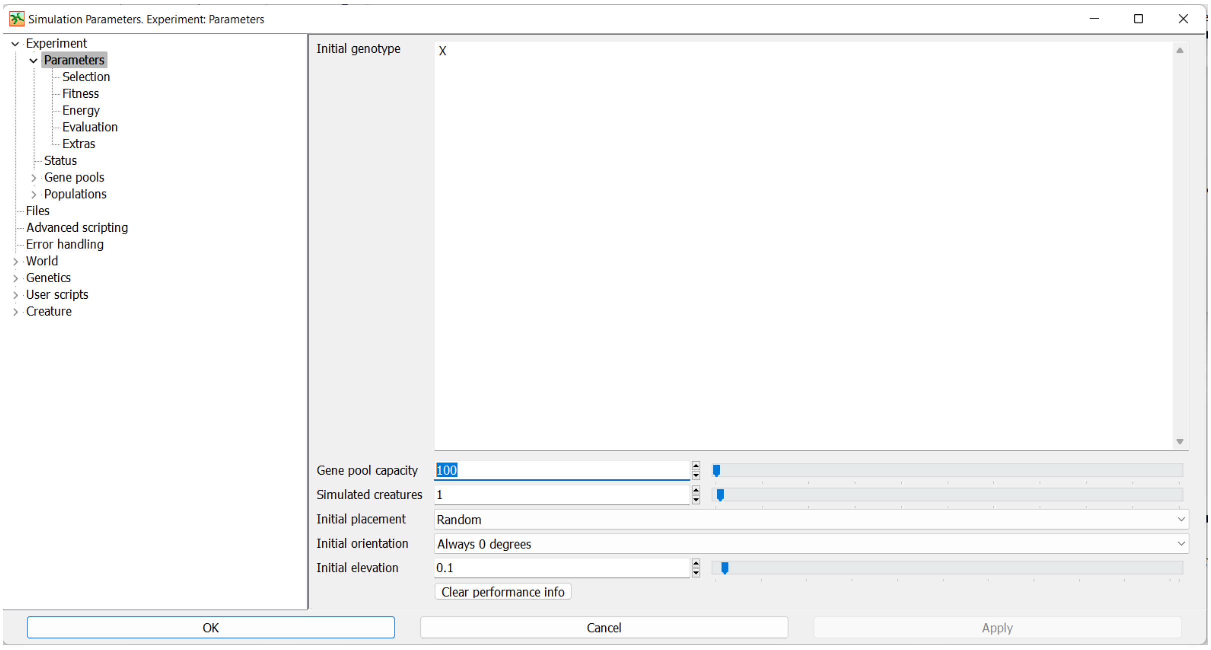
Think about what you want to achieve. It will define your fitness function. There are a lot of different possibilities but I suggest choosing something that can be easily seen during the visualisation of the creature (height, number of elements, velocity, desired movement type like jumping etc.) since it will probably look better on your report.

2. Specify the parameters

This is the most crucial thing that will significantly affect the experiment's output. In this section, I will go through different possibilities and share some problems I encountered during my work.

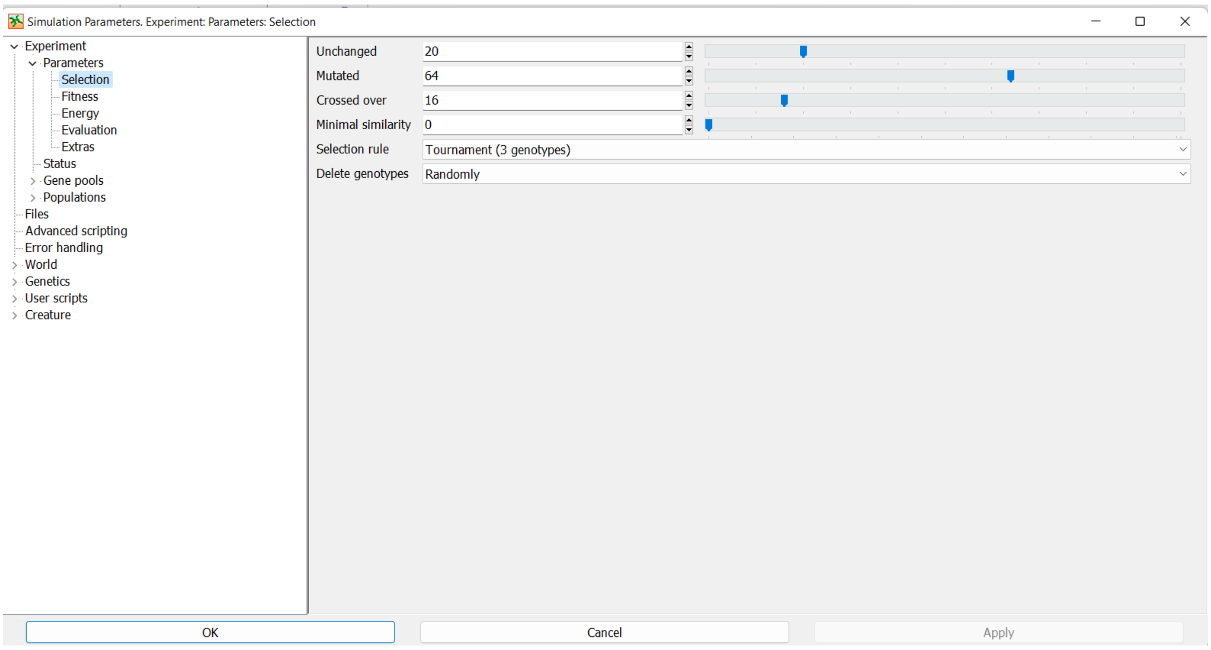
(The values on the screenshots are default ones)

The main **PARAMETERS** menu:



Initial genotype: X
Gene pool capacity: 100
Simulated creatures: 1
Initial placement: Random
Initial orientation: Always 0 degrees
Initial elevation: 0.1

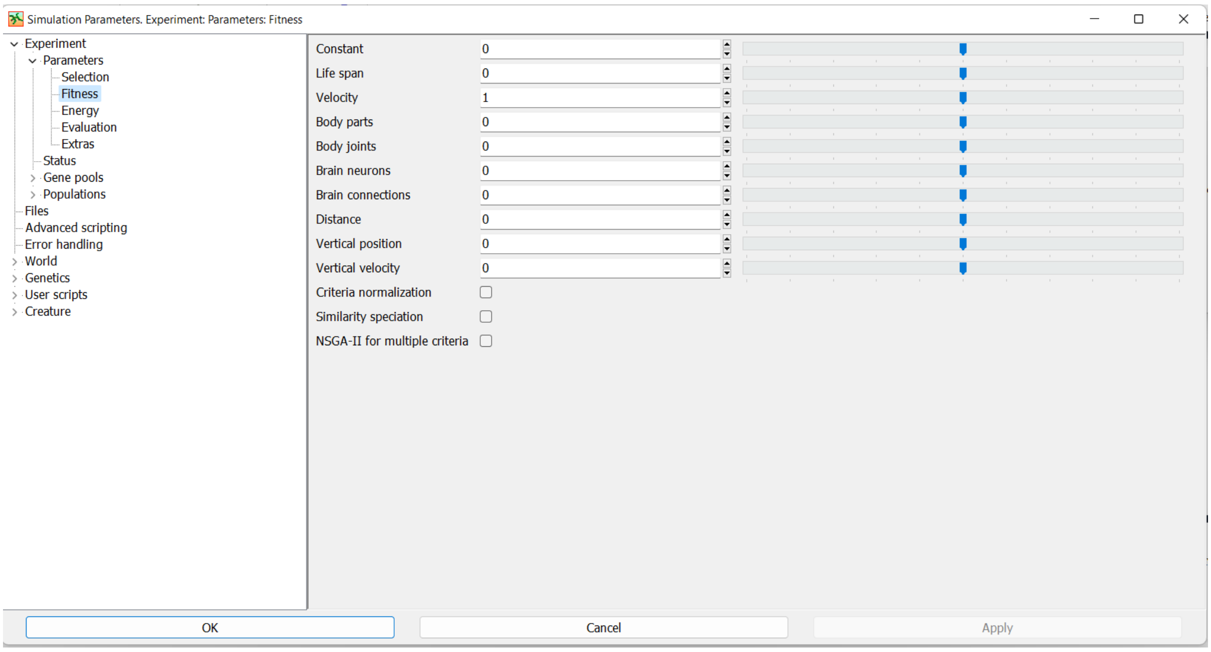
The only significant criterion on this page is gene pool capacity. It is equivalent to population size in other evolutionary algorithms. It can be beneficial to increase it in some instances but don't make it too big. Something around 100-200 will totally suffice.
The **SELECTION** menu:



Unchanged	20
Mutated	64
Crossed over	16
Minimal similarity	0
Selection rule	Tournament (3 genotypes)
Delete genotypes	Randomly

On this page, you specify how the selection will work. The default parameters aren't bad but I personally think that increasing mutation and decreasing crossover probability is a good idea. That way you will exploit the area faster which can be beneficial when you don't want to run your simulation for millions of iterations.
You can choose different selection rule e. g. roulette wheel (called fitness proportional here).

The **FITNESS** menu:



Constant	0
Life span	0
Velocity	1
Body parts	0
Body joints	0
Brain neurons	0
Brain connections	0
Distance	0
Vertical position	0
Vertical velocity	0
Criteria normalization	□
Similarity speciation	□
NSGA-II for multiple criteria	□

This is for sure the most important page. The values you specify here will affect which creatures will have a better chance of survival and reproduction and therefore will decide on the success or failure of the simulation in achieving the specified goal. Think about which attributes will encourage desired behaviour or other features of the creature.

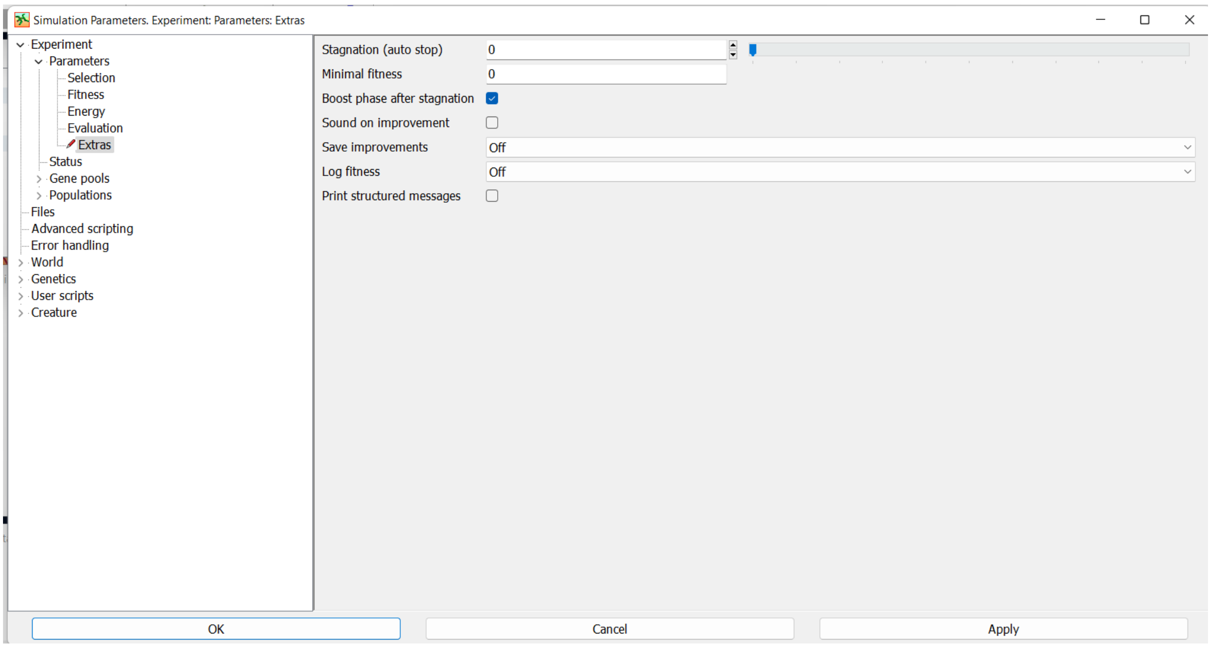
Don’t forget that different properties come in different orders of magnitude e.g. increasing the number of elements or vertical position by 1 is not such a big deal but if your velocities (horizontal or vertical) rose that much it would be a spectacular improvement (since it is represented as much smaller numbers like 0.001 for example).

It is very important while choosing the coefficients for different properties; it might not be a good idea to have a 1:1 proportion between velocity and vertical position in the fitness function.

How should you choose the coefficients exactly? It is hard to specify. The trial and error method will probably bring the best results; just experiment for a couple of tries and choose the one you are most satisfied with.

ENERGY and EVALUATION pages are not important for now

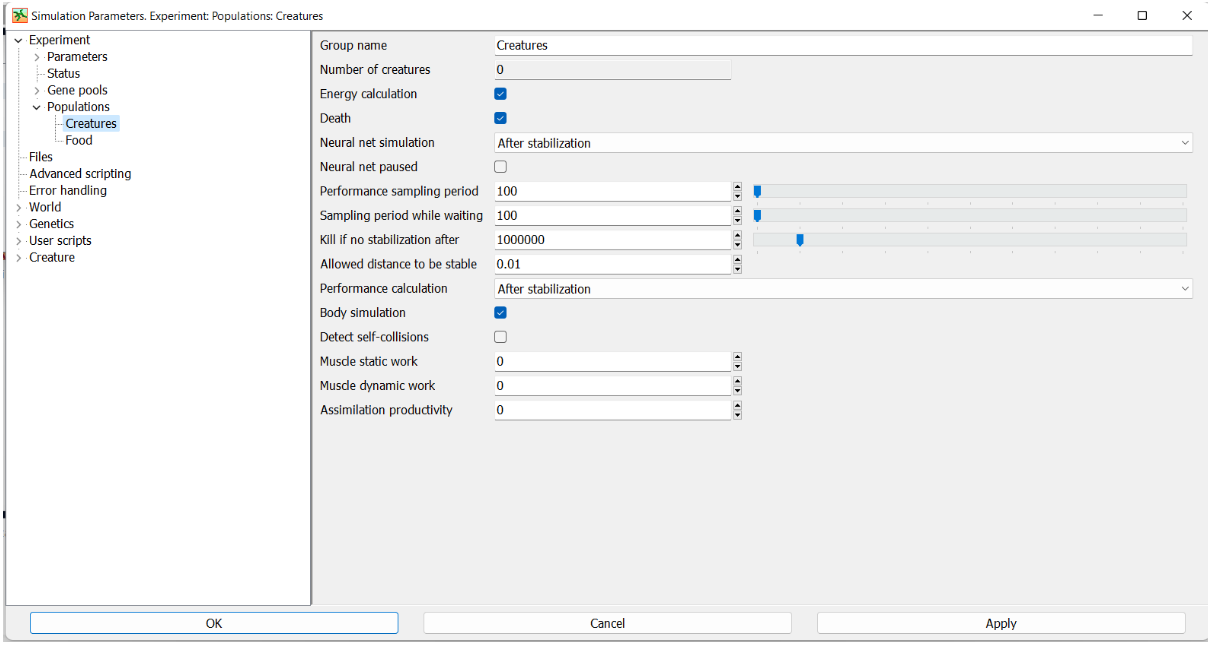
The **EXTRAS** menu:



Stagnation (auto stop): 0
Minimal fitness: 0
Boost phase after stagnation: ☑
Sound on improvement: □
Save improvements: Off
Log fitness: Off
Print structured messages: □

In this menu, you need to specify how many values should be saved to the log file. Don’t change anything besides ***LOG FITNESS***. Every 10 iterations seems like the best idea since saving every iteration slows the simulation considerably and to be honest, having so many values saved may make your work much harder when it comes to representing it in the form of the charts.

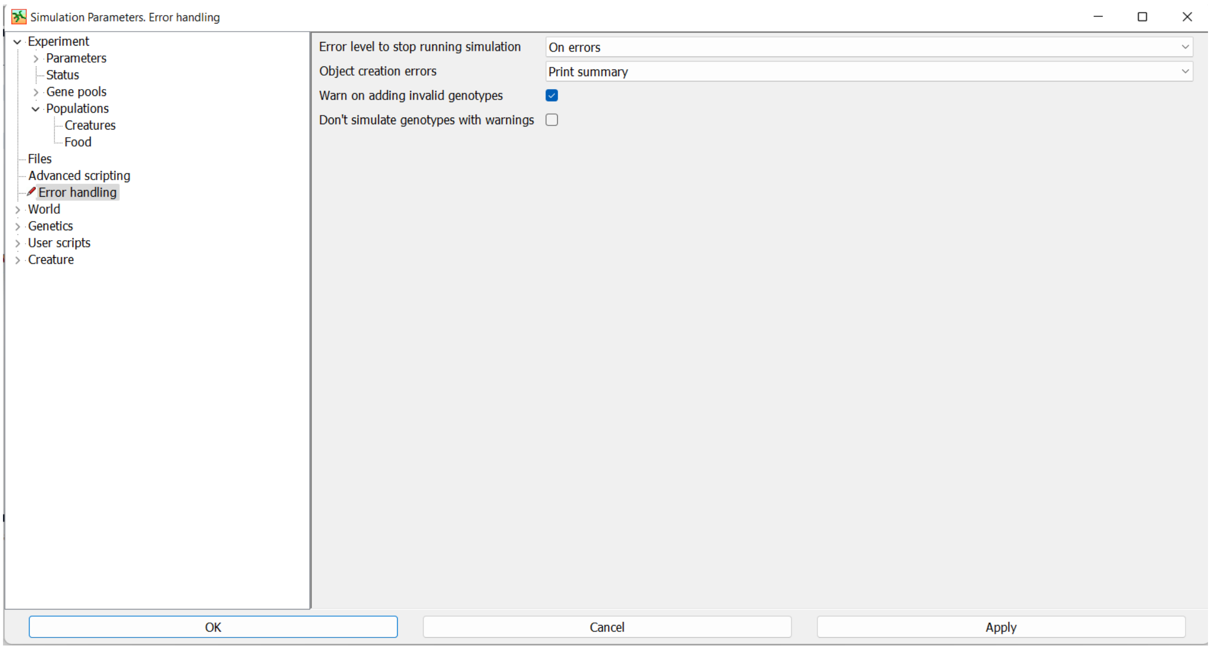
The **Populations -> Creatures** menu:



Group name: Creatures
Number of creatures: 0
Energy calculation: ☑
Death: ☑
Neural net simulation: After stabilization
Neural net paused: □
Performance sampling period: 100
Sampling period while waiting: 100
Kill if no stabilization after: 1000000
Allowed distance to be stable: 0.01
Performance calculation: After stabilization
Body simulation: ☑
Detect self-collisions: □
Muscle static work: 0
Muscle dynamic work: 0
Assimilation productivity: 0

The default values seem reasonable but you can play with them if you want to. The most crucial thing is for **DEATH** to be on since without it the creatures will be immortal and the evaluation of the single one will never end.

The **ERROR HANDLING** menu:

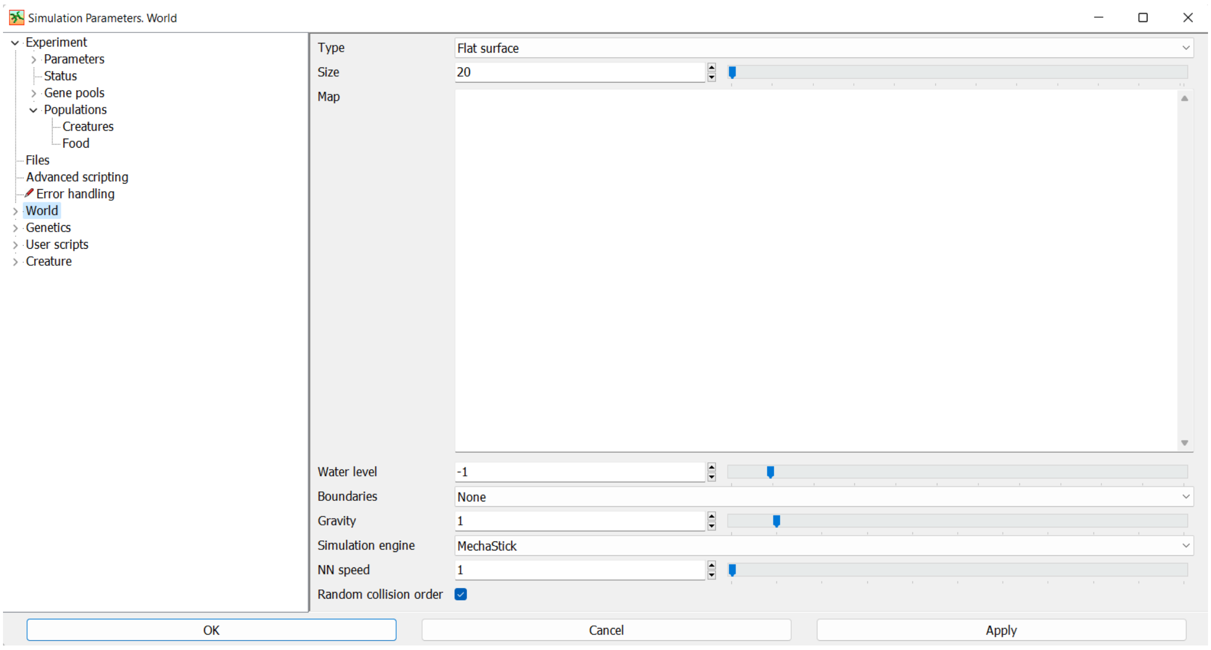


Error level to stop running simulation: On errors
Object creation errors: Print summary
Warn on adding invalid genotypes: ☑
Don't simulate genotypes with warnings: □

On this page make sure to turn the last option on. Without it, it is easy for your simulation to break by using incorrect (mostly wrong construction of the genotype) solutions then crossing them over and so on. This way most of your population can be “infected” and you will not achieve the desired results.

You can also decide not to print errors to the terminal but it is entirely up to you.

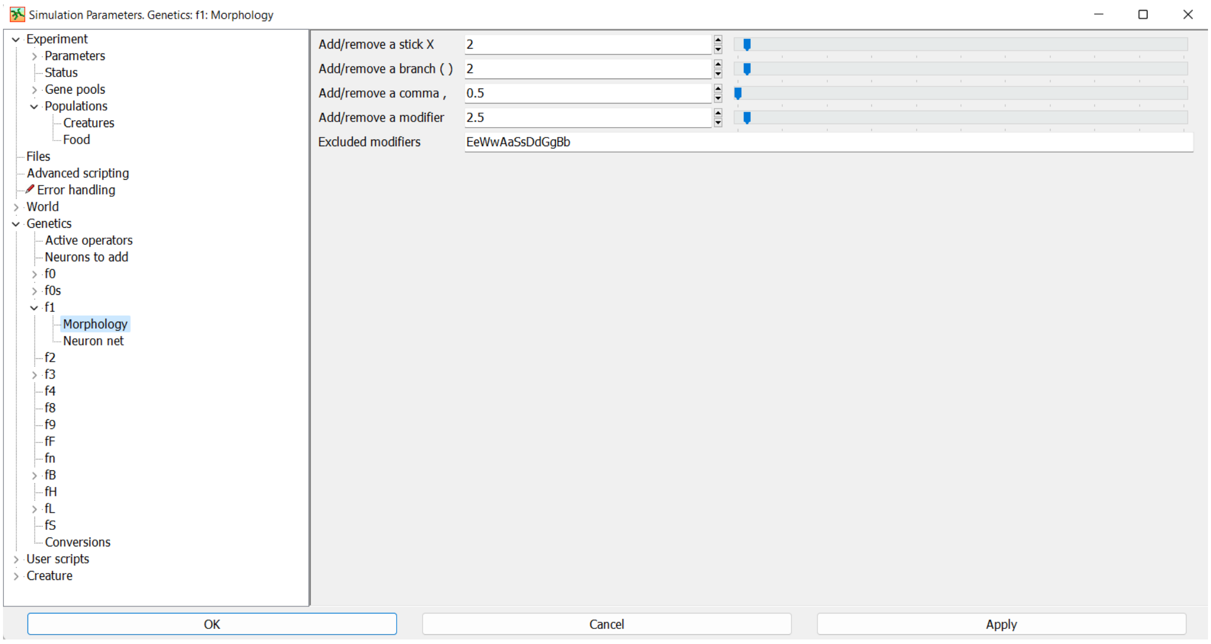
The **WORLD** menu:



Type: Flat surface
Size: 20
Map
Water level: -1
Boundaries: None
Gravity: 1
Simulation engine: MechaStick
NN speed: 1
Random collision order: ☑

Whether this menu is important or not depends on your initial goal. Do you want to simulate creatures on land or in water? Do you want to change the gravity or not? You can also change the used simulation engine but I don’t really see why you would like to use ODE for this report. I didn’t personally get into it so if you want to do something like that you need to explore it on your own.

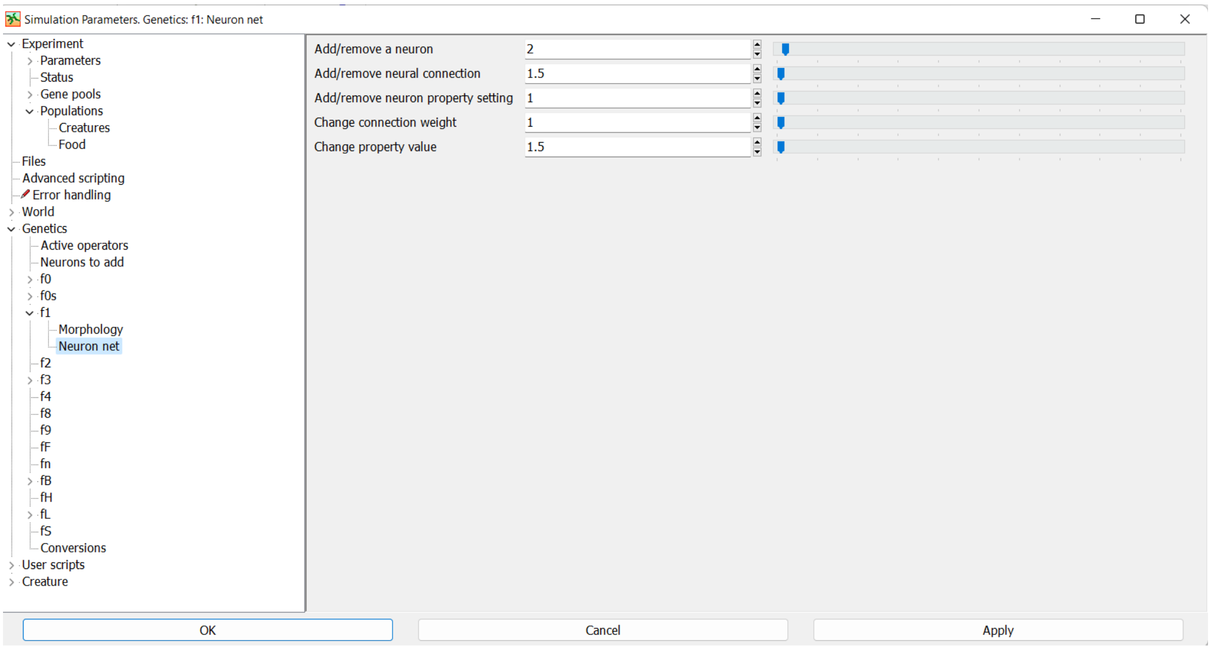
The **GENETICS** menu:



Add/remove a stick X	2
Add/remove a branch ( )	2
Add/remove a comma ,	0.5
Add/remove a modifier	2.5
Excluded modifiers	EeWwAaSsDdGgBb

Personally, I used f1 encoding and don’t know much about others therefore I will focus only on f1.

I think that default values are ok for longer simulations but with limited resources (mostly computational power and limitations of the software) they are far too small. If you want to achieve significant progress faster you need to increase it much. For morphology, I personally used values in the range of 20-40 (mostly the higher ones). With that even after 30000 iterations, the evolutionary progress was clearly visible.



Add/remove a neuron	2
Add/remove neural connection	1.5
Add/remove neuron property setting	1
Change connection weight	1
Change property value	1.5

For **NEURON SETS** I used the values of 20 for each of the parameters but you can use something else.

After changing the parameters save your settings as fileName.sim in the data folder

3. Performing the experiments

Encountered problems:

LOGS

The most important thing is that the version of Framsticks I worked with was broken and didn’t provide the functionality of saving the values to the log file. Therefore I had to change the experiment script by adding this functionality. It is not by any means perfect but it works.

You can find the modified version of the script in this repository.

Just download it and replace it in **framsticks/data/scripts**.

The main thing is that you have to manually create the log.txt file in

framsticks/data/scripts_output

The values from each run of the experiment will be added to the file therefore it can get quite big and there is a need to “clean” it every time you start a new set of runs. Make sure to save logs from each modification in separate files; it will be much easier to process.

BATCH

Running simulations manually can be quite boring and take a lot of time. Therefore it is a good idea to use a batch script to automize it. Copy the following code and save it in the folder in which you have installed Framsticks with the .bat extension.

```
FOR /L %%N IN (1,1,10) DO (frams "im fileName.sim" "Math.randomize();"
"Simulator.init();" "Simulator.start();" "while
(ExpState.totaltestedcr<30000) Simulator.step();" "sa fileName%%N.expt"
"-q")
```

Change fileName to the actual name of your .sim file. You can also modify the number of tested creatures. I personally liked 30000 for experimenting. It provided quite fast and representative results.

Remember that running your simulation only once is not sufficient. Since evolution is random some runs will be significantly worse than others. Run the simulation a couple of times and then decide if you are satisfied with the results.

4. Tips and Tricks

GOAL SELECTION

It is likely that the simulation will not satisfy your initial goal. Sometimes bad fitness function is responsible but not always. For example, I wanted to evolve fast and high giraffe-like creatures but it turned out that it is not really possible with the time I've got and the limitations of the program. The physical engine prefers jumping rather than walking creatures when optimizing for vertical position and velocity (since it is hard to evolve a high walking structure without additional constraints).
Don't be afraid to change the goal if you're not satisfied with the results.

CHARTS

When you will perform your experiments and decide on the final fitness function it might be a good idea to increase the number of evaluated creatures to several hundred thousand. Firstly longer evolution will probably lead to better results and which is not less important will look much better on the charts.

FITNESS FUNCTION VALUES

When constructing fitness functions make sure to make the possible values not much distant from each other. If you start at 0 and reach 200 by 1000 and 500 by 10000 iteration the chart will look absurd. Since normalization is not really an option (since it has to be easy to compare the values between different modifications of the experiment) designing your function in a way that doesn't produce such a difference is highly advised.

If you already have your function and encounter this problem don't worry. Just divide all coefficients by some value. After all the proportions between them are what really matters.

MODIFICATION IDEAS

The most trivial modification ideas are:

- Changing the Genepool size
- Changing mutation, crossover and unchanged probabilities in the **SELECTION** menu