Neural Eaters

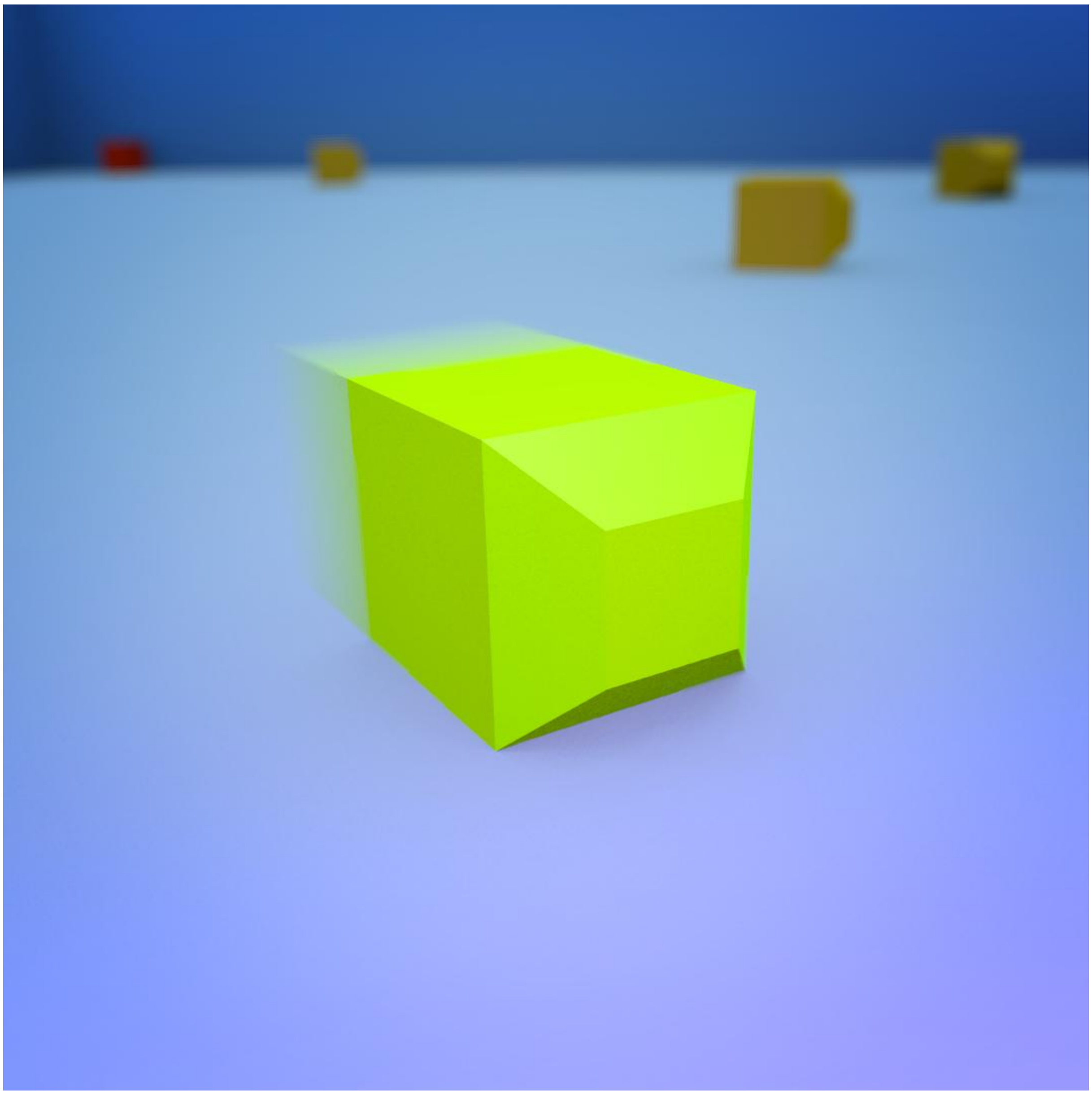

Table of content

What is it?

Neural Eaters is an evolutionary neural network simulation. The goal is for the Eaters, or entities, to learn how to find and eat the food. When they find the food they have a choice if they should reveal the foods location to everyone, or keep it to themselves. The chance of an entity to mutate to be able to find the food is relatively small, and other behaviors, like going in circles, will be far more popular. The amount of energy they have is visualized by their color.

The in game time is not constant and differs from real time. This is because the game is ran at the maximum speed that the computer can handle. Unfortunately the only way of changing this is by going into the .blend file and making "Use Frame Rate" true.

Every entity has its own brain, consisting of a number of layers. Every layer has a number of neurons. A neuron has two inputs, one mathematical operation, and one output. The inputs can either be a bias (a constant number) or an output of the previous layer. The neuron applies the mathematical operation and makes it the output. The first layer's neurons acquires their inputs from the entity's own inputs. The last layer's neurons outputs the entity's movement, rotation and if it tells other entities when it has found food.

Every entity also has its own energy. Eating food gives it more energy, while doing nothing or touching the wall will remove energy. The food is a purple pyramid that spawn randomly. When an entity has 0 energy it dies and is replaced by a new one. The game then also randomly selects an entity with a high amount of energy and then copies and mutates its brain. The chosen entity is visualized with a red arrow. This new mutated brain is applied to the newly created entity.

This process favors the entities which eat food and kills the ones that don't. In some sense it imitates the natural selection process in nature. It creates an artificial evolution.

Controls

Running

Arrow keys - Navigate player entity.

Tab - Show brain (Only in camera 1)

Tab + **Ctrl** - Show brain function

Space - Select and Deselect current latest mutated (Selection affects camera 2 and 3, brain visualization and sidebar information)

Space while in game - Reveal food.

1 - Camera 1, goes to default view. Keys I, J, K and L can be used to move camera and keys O and P to zoom.

2 - Camera 2, follow entity. Zoom dependent on vision.

3 - Camera 3, selected point of view.

4 - Camera 4, full map view.

S - Save to file. (Only in downloaded version.)

E and **R** - Change mutation

D and **F** - Change vision

Menu

Up and **down arrow** - Go up/down

Left and **right arrow** - Scroll integer up/down or change Boolean

Space - Select

Menu settings

Entities

Count - Number of entities.

Vision - How far every entity can see

Start energy - How much energy entities have when spawned.

Custom - Number of entities with custom brains.

Custom mutation - Entities are mutated at initial spawn.

Brain

Neurons - Approximate number of neurons in each brain.

Layers - Number of layers in each brain.

Memory - Number of Memory nodes. (Output becomes input next frame.)

Mutation

Mutation rate - Amount and chance of mutation

Lifetime - Smaller mutation the longer entity has lived. (Overwrites Mutation rate.)

Session

Load File - save.pk file is loaded at initiation.

Game - Player competes against entities.

Log - Save log to log_plot.lgdat.

Log Frequency - How often data is logged in seconds.

Log Session - Data is logged to Data.dat instead.

Log Session length - How long the data is logged to Data.dat.

Map
Size - Size of map.

Inputs
Ray - If ray cast hits wall, 0.01 is added every frame. If not it's 0.

foodDist - How far away the food is. (If Entity knows about it.)

foodRot - If entity is angled directly away from food, 0. If angled to food slightly to the right -180 and if slightly to the left 180. (If entity knows about it.)

otherNear - Number of other entities in vision.

Energy - Own energy.

Other

Reward and punishment
To help the entities evolve I have put in multiple rewards and punishments in the form of giving and taking away energy.

The first and most fundamental one is that a small amount of energy always is taken away. This is necessary for the simulation to function because else, no entities would die. The amount of energy taken is relative to how much energy the whole group has. If they share the food, the total group energy is high and less energy is taken.

They are rewarded for moving forwards, to make them move. They also get some energy when another entity dies.

Outputs
Move - If positive, move forwards. If negative, move backwards.

Rotation - If positive, turn left. If negative, turn right.

Alert - If 1 or above, alert others about food.

Saves and custom brains

When the "Load File" setting is true the game looks in the "save.pk" file and loads all entities with their brains, position, rotation and energy. Also the food size and position are loaded. Note that inputs are not saved in custom brains, and that you therefore have to use the correct input settings for the entities brains to function as intended.

To copy a brain from a save file to the custom brain file, you first have to identify which brain it is by taking note of its id while running. The id is presented in the sidebar info. In the save file the id is written on the same line as "NewEnt". After finding the right entity in the save, copy from the first following "NewLay" to the line before the next "NewEnt" or the end of the file. Then paste it into the "Custom.pk" and make the last line "End".

Logging

It is possible to log the average amount of energy to a file while running. This by changing the "Log" setting to true. When on, the game will append the average energy as a new line in the file every 1 second. The frequency of the logging can be changed.

The log file is named "Log_plot.lgdat". LiveGraph is what I'm using to graph the data, although other applications likely functions just as well.

If both the "Log" and "Log Session" settings are true, the game will log the data to the "Data.dat" file instead. Logging will only occur for the amount of seconds specified by the "Log Session Length" setting. Running multiple sessions with this true will not overwrite, but rather append the new data. You can then run the "combineData" file, which will combine all the logged data from the different sessions and write it to "Log_plot.lgdat". For everything to function correctly with "Log Session" you have to run every new sessions with the same frequency ("Log Freq.") and time ("L. S. length").