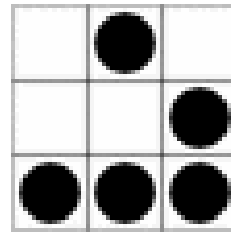


PROJET PERSONNEL ET DE FORMATION

2000 – 2010 et 2020+

Modélisation et simulation biologique

GABRIEL CHANDESRIS



BioInSilico – Modélisation de systèmes biologiques

Vie Artificielle : conception, développement, utilisation....

Notes personnelles, cahier des charges, idées, contraintes...

Table des matières

Remerciements	1
Résumé	2
Introduction	3
1 «État de l'art», notes et inspirations	4
1.1 Le jeu de la vie – <i>Life Game</i>	4
1.1.1 Les automates cellulaires	4
1.1.2 Le jeu de la vie	4
1.2 Modèle mathématique d'écologie	4
1.3 Les virus informatiques	5
1.4 Le jeu <i>Creatures</i>	5
1.4.1 Génétique des norns	6
1.4.2 Outils autour de <i>Creatures</i> «1.0»	6
1.4.3 Évolution du jeu : <i>Creatures</i> , 2, 3, <i>Docking Station</i>	7
2 Special notes about Genetics of Creatures	8
2.1 Genetics	8
2.1.1 Genes	8
2.1.2 Reference Tables	8
2.2 Brain Lobes	8
2.2.1 Overview	8
2.2.2 Brain Lobe Genetics - Genes' Header	9
2.2.3 Cell Body Page	10
2.2.4 D0 Growth Page	11
2.2.5 D0 Dynamics Page	12
2.2.6 Dendrite Overview	14
2.3 Chemical Emitter	16
2.3.1 Overview	16
2.3.2 Dialog Box	16
2.3.3 Breakdown – <i>Gene Header</i>	16
2.4 Chemical Receptor	17
2.4.1 Overview	17
2.4.2 Dialog Box	17
2.4.3 Breakdown	17
2.5 Initial Concentrations	18
2.5.1 Overview	18
2.5.2 Dialog Box	18
2.5.3 Breakdown	18
2.6 Creature Instinct	18
2.6.1 Overview	18
2.6.2 Dialog Box	19
2.6.3 Breakdown	19
2.6.4 Notes	20
2.7 Creature Stimulus	20
2.7.1 Overview	20
2.7.2 Dialog Box	20
2.7.3 Breakdown	21
2.8 Brain Lobes – <i>Creatures 1</i>	22
2.8.1 Overview	22
2.8.2 Lobe 0 – Perception Lobe	23
2.8.3 Lobe 1 – Drive Lobe	24

2.8.4	Lobe 2 – Stimulus Source Lobe	24
2.8.5	Lobe 3 – Verb Lobe	26
2.8.6	Lobe 4 – Noun Lobe	26
2.8.7	Lobe 5 – General Sensory Lobe	26
2.8.8	Lobe 6 – Decision Lobe	27
2.8.9	Lobe 7 – Attention Lobe	28
2.8.10	Lobe 8 – Concept Lobe	28
2.8.11	Lobe 9 – Regulator Lobe	29
2.8.12	What are Dendrites?	30
3	Idées générales – «Discussion»	31
3.1	Définition de règles de bases de conception	31
3.1.1	Quelques indications générales	31
3.1.2	Encodage de l'Information Génétique	31
3.1.3	Utilisation du «multithreading»	33
3.2	Idées d'intégration dans le modèle	33
3.2.1	Agents infectieux : bactéries, virus et plasmides	33
3.2.2	Système immunitaire	34
3.2.3	Étude évolutive	34
3.2.4	Gènes particuliers	34
3.2.5	Reproduction asexué, sexuée...	35
3.2.6	Parthénogénèse – <i>Extraits de Wikipedia</i> [41]	35
3.2.7	Biochimie et voies métaboliques	36
3.3	Mise en place de la programmation	39
3.3.1	Représentation objet	39
3.3.2	«Chimie» – métabolisme et génétique	39
3.3.3	Interactions entre l'individu et son environnement	39
3.3.4	Types d'agents : automates, plantes, bactéries, fourmis...	40
3.3.5	Codes de transcription, traduction...	40
4	Développement	41
4.1	Variables, types de gènes... Notes d'implémentation.	41
4.1.1	<i>Chemicals</i> – Variables	41
4.1.2	<i>GeneHeader</i> – En-tête	41
4.1.3	<i>InitialConcentration</i> – Concentrations initiales	41
4.1.4	<i>BiochemicalReaction</i> – Réactions Biochimiques	41
4.1.5	<i>BrainGene</i> – Cerveau	41
4.1.6	<i>BrainLobeGene</i> – Lobes du cerveau	41
4.1.7	<i>EmitterReceptor</i> – Émetteurs et récepteurs	42
4.1.8	<i>StimulusDecision</i> – Perception et décision	42
4.1.9	<i>Instinct</i> – Réseaux neuronaux pré-établis	42
4.2	Liste normalisée des «éléments chimiques»	43
4.3	Encodage génétique	45
4.3.1	Gènes «paramètres» – <i>GeneGattaca</i>	45
4.3.2	Gènes «encodage complet»	46
4.4	Modélisation et conception d'agents – Construction de tests	49
4.4.1	Entretien de l'environnement : les <i>Daemons</i>	49
4.4.2	Organismes de bases : bactéries et plantes	49
4.4.3	Agent en déplacement	49
4.4.4	Espèces sexuées	49
4.4.5	Virus, système immunitaire : impact...	49
4.4.6	Écosystème complexe	49
4.4.7	Modèle complexe : «Test de la fourmilière»	50
4.5	Interfaces Homme-Machine	51
4.5.1	Conception de gènes et voies métaboliques	51



TABLE DES MATIÈRES	iii
4.5.2 Conception d'organismes (génétique)	52
4.5.3 Conception avancée d'organismes (phylogénie et documentation)	52
4.5.4 Autres outils (interaction)	53
5 Reprise du projet en 2020	54
5.1 Sortie de sommeil du projet	54
5.2 Publication GitHub et autres idées	54
Abbréviations	56
Bibliographie	57

Table des figures

1	Une cellule et ses voisines	4
2	General Interface for Brain Genetics Design	9
3	Cell Body Interface for Brain Genetics Design	10
4	D0 growth Interface for Brain Genetics Design	11
5	D0 dynamics Interface for Brain Genetics Design	12
6	Interface for Chemical Emitter Design	16
7	Interface for Chemical Receptor Design	17
8	Interface for Initial Concentration Design	18
9	Interface for Intincts Design	19
10	Interface for Stimulus Design	20
11	Creatures 1 Brain Map [12]	23
12	Creatures 2 Brain Map [12]	23
13	Tableau de Mendeleïev.	37
14	Étapes de la glycolyse.	37
15	Schéma de déroulement du cycle de Krebs.	38
16	Formule descriptive du gène <i>BiochemicalReaction</i>	41

Liste des tableaux

1	Lobes and perceptible settings	19
2	Norn Lobes for <i>Creatures 1</i>	22
3	Norn Lobes for <i>Creatures 2</i>	23
4	Copied Lobes in Perception	23
5	Drive Lobe Data Entries <i>Drives</i>	24
6	Source Lobe Data Entries	25
7	Verb Lobe Data Entries	26
8	General Senses Lobe Data Entries	27
9	Regulator Lobe Loopbacks	30
10	Tableau des 64 triplets standards du code génétique	31
11	Expérimentation de base avec une bactérie de test.	33
12	Nomenclatures des acides gras saturés linéaires de 1 à 32 carbones	38
13	Nomenclature des variables de chimie et biochimie	44
14	Nomenclature pour les Stimuli et Décisions	44
15	Les 64 triplets du codage paramétrique <i>GeneGattaca</i>	45
16	Description brève des types de gènes paramétriques	47
17	Tableau des 256 quadruplets standards du codage «complet»	48
18	L'interface <i>GeneCreator</i>	51
19	L'interface <i>GeneticKit</i>	52
20	L'interface <i>OrganismKit</i>	52



Remerciements

Résumé



Introduction

[...]

1 «État de l'art», notes et inspirations

1.1 Le jeu de la vie – *Life Game*

1.1.1 Les automates cellulaires

Les automates cellulaires constituent une classe de systèmes dans laquelle une grille de cellules évolue de cycle en cycle en fonction de règles définies au niveau de chaque cellule et spécifiant quel doit être l'état de la cellule au cycle suivant en fonction de son état courant et de l'état de ses plus proches voisines. À chaque cycle, toutes les cellules recalculent leur nouvel état puis elles changent toutes d'état simultanément, de manière synchrone.

1.1.2 Le jeu de la vie

Le jeu de la vie, proposé par J. Conway dans les années 1970 est l'exemple le plus connu et le plus étudié de ce type de systèmes. Cette simulation se déroule sur une grille à deux dimensions, théoriquement infinie (mais de longueur et de largeur finies et plus ou moins grandes dans la pratique), dont les cases — qu'on appelle des «cellules», par analogie avec les cellules vivantes — peuvent prendre deux états distincts : «vivantes» ou «mortes». Chaque cellule ne peut se trouver à chaque instant que dans l'un des 2 états **on** (allumée, vivante) ou **off** (éteinte, morte). Lorsqu'elle est allumée, la cellule ne restera allumée au cycle suivant que si 2 ou 3 de ses 8 voisines sont elles-mêmes allumées. Lorsqu'elle est éteinte, la cellule ne s'allume au cycle suivant que si exactement 3 de ses 8 voisines sont allumées.

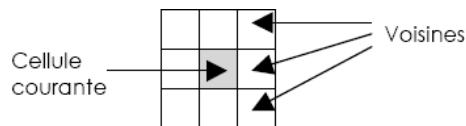


FIGURE 1 – Une cellule et ses voisines

D'autres variantes de ces règles sont définissables, dans la mesure où chaque cellule connaît l'état de ses huit voisines. Il suffit de faire varier le nombre de cellules vivantes nécessaires pour en faire naître une nouvelle, la maintenir en vie ou la faire mourir. Il faut préciser que le jeu de la vie n'est pas vraiment un jeu au sens ludique, puisqu'il ne nécessite aucun joueur ; il s'agit d'un automate cellulaire, un modèle où chaque état conduit mécaniquement à l'état suivant à partir de règles pré-établies.

Une étude systématique de ce type de modèle a été effectuée par Stephen Wolfram au de 1980 à 2003 [42].

1.2 Modèle mathématique d'écologie

Des modèles plus complexes ont été construits, toujours sur ce même principe d'un ensemble de règles mathématiques ou physiques simples et définissant le déroulement de la simulation et sa paramétrisation [25, 24, 11] : par exemple pour étudier le gradient d'un produit ou présence d'un autre organisme sur la survie d'organismes modélisés. Chaque «bactérie» ne peut avoir qu'un nombre limité de «concurrents» (qualité implicite d'un milieu renouvelé), gradient entre la surface et le fond de la «boîte de Petri»...

Un cycle de calcul dans ce type de modélisation peut se décomposer de la façon suivante :

1. Reproduction
2. Mutation
3. Mouvement
4. Compétition
5. Sélection



1.3 Les virus informatiques

Mark A. Ludwig *Mutation d'un virus*

- Virus qui favorise l'évolution
- Mutations quasi "aléatoires"
- Recherche du code des survivants
- Mécanismes de reproduction
 - Modules inclus au virus
 - Manipulation d'un élément

Un même virus peut posséder plusieurs fois le même moteur en copie interne : sélection d'une partie du code et la copie est ajoutée à la fin.

Il y a dans ces exemples utilisation d'un moteur darwinien de sélection génétique : connaissant le risque de se faire prendre à chaque génération (utilisateur, antivirus), donc si le virus en réchappe une fois, il faut pouvoir continuer, d'où une conservation de son passé «génétique» et de son développement. Il y a donc ici une exploitation des informations génétiques, associé à un souvenir des mutations et la conservation des propriétés parentales.

Différents moteurs de fonctionnement

- reproduction
- mutation
- analyse
- destruction
- contrôle
- mémoire éventuelle

1.4 Le jeu *Creatures*

La première version grand public du jeu *Creatures* est disponible en 1996 (pour Windows 95 et Mac OS Classic) ; l'objectif principal de ce modèle de simulation est la modélisation de comportements adaptatifs en utilisation la vie artificielle [8, 14, 15, 16, 17]. Il ne s'agit plus vraiment ici d'automates cellulaires, les «individus» simulés sont appelés des **Norns** (*Cyberlifogenis cutis*), et sont accompagnés dans leur environnement virtuels par des Grendels (*Cyberlifogenis vicious*) et des Ettins (*Cyberlifogenis kleptomonicus*).

La simulation est fortement centrée autour du système nerveux des norns, les neurones sont organisés en groupes (lobes, la terminologie biologique est reprise). Différents paramètres définissent les changements d'états des neurones, ces paramètres sont identiques pour les neurones au sein d'un même lobe. Certains paramètres des neurones sont des règles de calcul de l'état (excitation, repos, relaxation...) ou de connexion avec d'autres neurones au sein du même lobe ou vers d'autres lobes. Autour de ce modèle de réseau neuronal, une biochimie simple est simulée et codée génétiquement pour permettre une gestion des informations en fonction d'un système hormonal (endocrinien) et d'un métabolisme basique.

Lors de la création des commandes et fonctions associées aux gènes il a été convenu de déterminer les variables et les écarts de valeurs numériques possibles. Pour certains types de gènes cette valeur détermine son (in)activité (0 ou 1) et peut dépendre de l'âge de la vie du norn, d'autres sont fixes pour sa vie (apparence/immunité sauf par accident ou de façon volontaire, comprises entre 1 et 26), et donne lieu qu'à l'expression d'un seul type de caractère (ici une lettre qui détermine l'identité du norn).

Enfin certains gènes ont une variable dont le niveau détermine l'activité du gène (ex : gènes des lobes du cerveau, le niveau de la variable détermine dans quel domaine il exerce son activité : sensoriel, conceptuel, décision, attention...) pour ce dernier type de gène la tolérance pour la valeur de la variable est large et les "créneaux" où l'activité est déterminée sont larges pour éviter qu'une mutation ponctuelle ne modifie immédiatement l'activité du gène.

Les mutations s'effectuent surtout lors de la reproduction («erreurs de transcriptions») et peuvent modifier les valeurs de différentes façons : changement de valeur (0/1), incrémentation ou décrémentation (apparence, immunité et lobes), duplication/élimination du gène. La probabilité de mutation est la même pour chaque gène (le processus recommence à chaque fois). En de rares événements, des mutations peuvent se produire suite à une infection (agents pathogènes ou autre) ou l'exposition à des «radiations».

1.4.1 Génétique des norns

Le génome des norn est haploïde (un seul chromosome) [14]. Lors de la reproduction, les gènes peuvent être dupliqués, mutés voire délétés, avec certaines limitations pour certains gènes et vérifier la viabilité (ou la non-viabilité) d'un nouvel individu ; un crossing-over peut également être appliqué entre les gènes parentaux. Chaque gène possède un en-tête pour indiquer s'il peut être muté, dupliqué ou délété (chaque critère n'est pas exclusif ou limitatif pour les autres), ainsi que l'activation du gène selon l'âge.

Différents types de gènes sont définis dans *Creatures* [14] :

- **Chemicals // InitialConcentration** ou éléments chimiques : ce sont des variables définies génétiquement indiquant des concentrations de produits dans l'organisme du norn, sans aucune définition précise des propriétés des produits. *Chaque gène de ce type indique les numéros des produits, leur nom, leur concentration, leur durée de demi-vie...*
- **Emitters** : ces éléments sont produits par des objets émetteurs (définis génétiquement) et peuvent être reliés à des sorties du réseau de neurones. Le changement de valeur de l'élément relié provoque un changement de sortie de l'émetteur. *Les gènes de ce type contiennent le "lieu" d'émission, le numéro du produit et le fonctionnement de l'émetteur.*
- **Receptors** : ces éléments ont le fonctionnement opposé des récepteurs, ce sont des entrées du système nerveux. *Les gènes des récepteurs contiennent le numéro du produit auquel réagir, le niveau à partir duquel régir et le signal d'entrée.*
- **Reactions** : les réactions chimiques sont le centre du métabolisme des norns, et peuvent avoir une large palette de comportements :
 - $A + B \rightarrow C + D$: réaction chimique standard
 - $A + B \rightarrow C$: fusion de réactifs
 - $A \rightarrow \emptyset$: diminution (l'inverse n'est pas permis)
 - $A + B \rightarrow A + C$: catalyse (A est inchangé)
 - $A + B \rightarrow A$: diminution catalytique de B*Les gènes définissant les réactions chimiques définissent les coefficients et la vitesse de réaction.*
- **Gènes d'apparence et du système nerveux** (incluant les Stimulus et les Instincts, qui ne seront pas détaillés ici, mais pour lesquels de nombreuses ressources sont disponibles [8, 14, 15, 16, 17]).
- De plus, des infections bactériennes des norns sont possibles, ils disposent d'un système immunitaire qui réagit aux antigènes bactériens par une production d'anticorps.

1.4.2 Outils autour de *Creatures* «1.0»

Les logiciels inclus à la simulation peuvent lire et modifier les différents fichiers de cette modélisation, les afficher et les interpréter pour l'utilisateur. Les fonctionnalités sont les suivantes :

- **CGM (Creatures GenMan) // Genetic Kit** : Analyse du génome d'un norn (gène par gène, affichage de ce à quoi correspond la variable de chaque gène), croisement entre deux génomes, introduction d'un nouveau norn avec un génome sélectionné et d'autres paramètres (stade de la vie, sexe...),
- **Object injector** : fait introduire des objets, permet la programmation de nouveaux objets (programmation simplifiée selon catégorie d'objets),
- **BORG** : Interaction directe avec le métabolisme interne des norns, sélection des objets et organismes non sélectionnables autrement au sein de la simulation (plantes, objets d'interactions comme des jouets...).

Chaque objet, présent dans l'environnement, est caractérisé par :

- Des variables (vitales, à ajouter, à soustraire selon le programme) suivant une liste normalisée.
- Une information génétique (ou programme spécifique de l'objet) (le code génétique peut être prévu de façon à autoriser la vie, mais aussi des opérations «simples» : augmenter une variable, diminuer une variable, faire un déplacement...).



1.4.3 Évolution du jeu : *Creatures*, 2, 3, *Docking Station*

Des versions ultérieures de ce jeu ont permis de faire évoluer différents aspects du jeu et du modèle sous-jacent. Les éléments qui ont évolué sont les graphismes, un système de feedback (lobe de régulation) dans la version 2, l'annotation du génome et son formalisme avancé dans la version 3, avec l'apparition des organes au sein des norns et la liaison des gènes avec ces organes. De plus, les règles de calcul du réseau de neurones ont évolué (les *State Variable Rules*) de façon à permettre la construction de règles plus élaborées.

Quelques notes de recherches expérimentales sur le fonctionnement et la documentation de *Creatures* sont présentées dans ce mémoire dans leur version originale (en anglais). Ces notes concernent la première et la seconde version du logiciel et proviennent notamment du *Creatures Developer Ressources* [7] et du site *geNornIcs* [12], disponibles respectivement aux adresses suivantes :

- <<http://www.double.co.nz/creatures/>> [7]
- <<http://meliweb.net/creatures/>> [12]

L'intérêt de cette section est de fournir une documentation existante sur une création réelle et de finaliser un état de l'art sur ce type de développement, au départ purement académique puis commercial dans le cas de *Creatures* ; un tel développement pouvant être repris pour modéliser un ensemble biologique complexe (organismes dans leur milieu, gènes, évolution...) et permettre à une communauté d'utilisateurs d'étudier un tel modèle et d'y ajouter diverses créations, de découvrir diverses mutations voire une évolution.

2 Special notes about Genetics of Creatures

2.1 Genetics

The behavior of a norn and its interaction with the environment within Creatures is controlled by its Digital DNA. This DNA code can be modified using the Creatures Genetics Kit. The Genetics Kit is the official means of editing a norns genome but there are various third party tools available as well. See the links page for other sites that may contain such utilities.

While this documentation concentrates mainly on the Norn brain lobes there are a number of other genes that are modifiable via the Genetics Kit. The online help in the Genetics Kit is not very helpful when it comes to descriptions of the individual fields within each gene dialog box. Follow the links below to get information on the various fields for each gene dialog in the Genetics Kit based upon information I've gathered on the net and experiments I've performed.

2.1.1 Genes

- Brain Lobe
- Chemical Receptor
- Chemical Emitter
- Chemical Reaction
- Chemical Half-Lives
- Chemical Initial Concentrations
- Stimulus
- Genus
- Appearance
- Pose
- Gait
- Instinct
- Pigment
- Pigment Bleed

2.1.2 Reference Tables

- State Variable Rules
- Cell List connection [12]
- Brain Map and Models [12]

The information presented here was written for Creatures 1 but all the information is still valid for Creatures 2 genetics. I have not yet updated the screenshots of the genetics kit for Creatures 2 but not much new was added that needs explanation.

2.2 Brain Lobes

The brain lobe is one of the most complicated portions of norn genetics. The Cyberlife Genetics Kit provides the ability to add, delete or modify brain lobes. The dialog box for modifying brain lobes contains a bewildering number of options. The purpose of this brain lobe discussion is to describe what each of these options does. Note 08/10/1999 : The information on brain lobes below was created for Creatures 1. All the information is still valid for Creatures 2 genomes, but the screen shots show the Creatures 1 genetics kit.

The descriptions of the brain lobe settings shown here have mostly been discovered through experimentation. The tutorials available in another section describes how most of the information was found. The purpose of this section is to provide a single place where all the information gathered through the tutorials and other places can be placed for easy use. It will be updated frequently as I discover new things.

2.2.1 Overview

Most standard norns have nine brains lobes. A number of the lobes are 'special' in that they have a particular function hard-coded within the Creatures executable. A limited amount of genetic modification can be done to these lobes as most of their functionality is provided through the Creatures system. The other lobes are completely genetically defined and this is where some of the most interesting modifications to a Norn can be made.



A brain lobe sits within a 64x48 grid of neurones (often referred to here as cells). A lobe has a location within this brain grid described as an (X, Y) coordinate and (width, height). Each lobe therefore has a set size measured in neurones. Each neurone in a lobe has a numeric value associated with it known as the 'state' represented as a number from between 0 and 255 inclusive. The current value of the 'state' of a neuron indicates the level of activity. The value of the neurone 'state' can be set via the Creatures executable directly, CAOS macro code, or via dendritic links from other cells in other brain lobes. In this manner a network of cells and lobes is created that ultimately forms the 'brain' or neural net of the norn.

All neurones within a single lobe behave in a similar manner to perform some function. Each lobe has genetically defined information associated with it that affects the neurones in that lobe. So a brain lobe is really a group of neurones that perform in a similar manner.

A detailed description of the genetically defined information associated with each lobe follows. This is the information that was obtained through experimentation. After this a brief description of the functionality of the brain lobes in a standard norn is provided. This document will conclude with some ideas and thoughts on future brain modifications that could be made to improve or modify a norns behavior.

2.2.2 Brain Lobe Genetics - Genes' Header

The Cyberlife Genetics Kit has a 'tabbed' dialog box for modifying the genetic information associated with a brain lobe. Each page of the dialog box holds information about the lobe and the breakdown below will be done page by page as defined by the genetics kit.

General Page

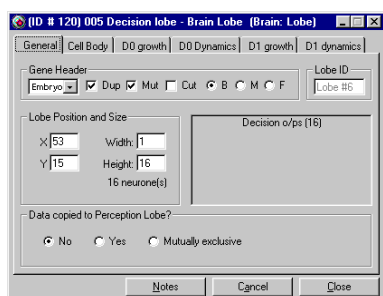


FIGURE 2 – General Interface for Brain Genetics Design

The example dialog given here is from the Decision lobe of a Ron norn. The 'general' page provides the standard gene header information that describes when the gene switches on, how mutations affect it and the required sex of the norn for the gene to switch on. Also included is information on the placement of the lobe within the brain grid, a description of the lobe, a unique numeric identifier for the lobe and an indication as to whether lobe data is copied to the perception

Gene Header

Embryo : This field indicates the age at which the brain lobe gene will switch on. Most brain lobes turn on at the 'Embryo' stage to ensure they operate correctly as soon as the norn is born. I know of no lobes that are turned on at different ages and I don't know if this functionality even works. This has been added to the list of issues to explore.

Dup : If this flag is checked then the lobe can be duplicated as a genetic mutation. That is, multiple copies of this lobe will be able to exist due to mutation.

Mut : Sets whether genetic mutations can be applied to the values contained within this brain lobe.

Cut : If set the brain lobe can be removed during breeding or as the result of mutation. Most brain lobes have this unchecked as a norn without any of the standard lobes would not be able to function.

B/M/F : Defines the gender that this brain lobe will switch on. This could potentially be used to create norns that have different intelligence levels or behaviour depending on the gender of the norn. I haven't tested if this functionality actually works. This has been added to the list of issues to explore.

Lobe Id

Lobe # : Contains the numeric identifier for the brain lobe. This is assigned automatically by the genetics kit and cannot be changed. The first 8 numbers (0 through to 7) correspond to specific lobes that are hard-coded within the Creatures executable. If you number a lobe with one of these hard coded numbers (using a hex editor or such like) then Creatures may stomp over your neurone values at any time so be warned.

Lobe Position and Size

X : Contains the X coordinate of the lobes location within the brain grid. I don't know what the effect of overlapping brain lobes will be. Added to the issues list.

Y : Contains the Y coordinate of the lobes location within the brain grid.

Width : Contains the amount of neurones in the X direction that the lobe uses within the brain grid.

Height : Contains the amount of neurones in the Y direction that the lobe uses within the brain grid.

The total number of neurones contained within the lobe is calculated as (Width * Height).

Description

This contains a description of the functionality of the brain lobe and is not editable. I think only the standard brain lobes have descriptions associated with them. All additional lobes get a default like 'Inter-neurone lobe'.

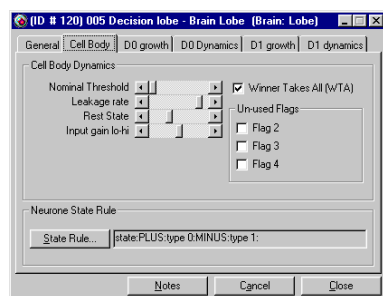
Data Copied to Perception Lobe

No : None of the information in this brain lobe will be duplicated in the perception lobe (lobe 0).

Yes : Every neurone within this brain lobe will have its state value duplicated in the perception lobe. See the perception tutorial for details.

Mutually Exclusive : A lobe marked as mutually exclusive affects the way dendrites are formed between the perception lobe and the concept lobe. If a lobe is marked as mutually exclusive then only one cell from this lobe may contribute to the forming of a particular concept in the concept lobe. See the perception tutorial for details.

2.2.3 Cell Body Page



The 'Cell Body' page provides information associated with each neurone (otherwise known as 'cell') within the brain lobe. It is here that the behaviour of the neurones are defined. Much of the information obtained about this page is from tutorial one and tutorial two.

FIGURE 3 – Cell Body Interface for Brain Genetics Design

Cell Body Dynamics

Nominal Threshold : This is a number with a value from 0 through to 255. When the state is set for a neurone it will not fire unless the value of the neurone state is greater than this threshold. The value of the neurone after firing will be state minus threshold.

Leakage Rate : This defines the speed at which the state will drop from its current value to its rest state. It has values like 'Instantly', '5 Seconds', right through to '52 Years' and is represented by a number from 0 to 255.

Rest State : This is the resting state of the neurone. That is, if the neurone has not been fired or activated it will sit on the value set here. The value is from 0 through to 255.

Winner Takes All : If this flag is checked then the lobe is a winner-takes-all lobe. This means that after all the neurone values for the lobe are calculated the highest firing neurone (highest state value) is the only one that fires and all the other neurones have their output value set to zero. An example is the decision lobe where only one decision can be made.



Neurone State Rule

An SVRule is like a miniature program written in a special programming language, which has a number of 'opcodes' or operations that it can perform on various pieces of data. Only eight individual opcodes are allowed in an SVRule making it very small and fast to execute - the SVRule for every cell in the brain must execute 10 times per second! – *State Rule* : The SVRule defined here is used to calculate the new state value of a neurone. The result of the SVRule becomes the state value of the neurone. In this way the state for neurones within lobes can be defined genetically and different calculations can be performed depending upon the task of the lobe. The SVRule possibilities are huge and can be of a great complexity. Remember that SVRules are used in a number of places in a brain lobe but the specific one here is only used to calculate the value of each neurone 'state'.

2.2.4 D0 Growth Page

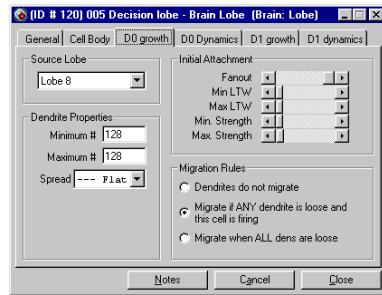


FIGURE 4 – D0 growth Interface for Brain Genetics Design

This is one of the first pages that describes the behaviour of dendrite connections in a lobe. Dendrites are to the brain lobes as wiring is to integrated circuits. They connect the neurones in different lobes and allow information to travel between them. Each brain lobe can have dendrite connections with up to two different source lobes. These are described as 'type0' dendrites and 'type1' dendrites. This page describes the settings for 'type0' dendrites.

Source Lobe : This defines the brain lobe these dendrites are connected to. Data flows from neurones in the source lobe defined here to neurones in this brain lobe.

Dendrite Properties

Minimum # : Defines the minimum number of dendrites that will be connected between each cell in the current lobe and a source lobe cell.

Maximum # : Defines the maximum number of dendrites that will be connected between each cell in the current lobe and a cell in the source.

- At birth a norn will have a random brain wiring pattern set up. Each cell will have a number of dendrites ranging between the minimum and maximum number defined here. As dendrites migrate the minimum and maximum numbers are important as they define whether there is room for a new dendrite connection to be made or removed. The decision on how many dendrites to use when creating a lobe depends upon the type of information to obtain from the source lobe and what to do with it.
- Spread** : This appears to define the pattern used to define the 'fanning' of the dendrite connection between the current lobe and the source lobe. The options are 'Flat', 'Normal', 'Saw' and 'waS'. If the spread is 'flat' then there will be two dendrites connecting neurone x from the source lobe to neurone x in the destination lobe, then (x+1) to (x+1), etc. With the other settings, a connection from neurone x in the current lobe to neurones x and (x+1) in the source lobe, from neurone (x+1) to neurones (x+1) and (x+2), etc. The exact layout of the dendrites will also depend on the 'fanout' value described later.

Initial Attachment

Fanout : When the dendrite wiring for a brain lobe is initially created on birth of a creature, this value defines how each dendrite connection from the current lobe 'fans out' to neurones in the source lobe. The value varies from 0 through to 8.

Min LTW : A number ranging from 0 through to 255 described as the minimum Long Term Weight. A dendrite has a value known as the long term weight (LTW) and it has a minimum and maximum possible value. It's actual value floats somewhere in between the minimum and maximum.

Max LTW : The maximum value of the Long Term Weight as a number from 0 through to 255. It must be equal to or greater than the min LTW.

Min Strength : Each dendrite has a strength value represented as a number from 0 through to 255. The strength represents how 'strong' the link between the source neurone and the current neurone. When the strength is '0' then the dendrite may migrate as the link is not strong. The minimum number defined here is used for the initial wiring of the brain lobe and for initial values after dendrite migration.

Max Strength : The maximum strength value that can be assigned to a dendrite on initial brain wiring.

Migration Rules

The radio button settings here define how dendrites will migrate between cells. An example of dendritic migration is the connections between the perception lobe and the concept lobe. The perception lobe is the source lobe and the concept lobe is the destination lobe. There may be several neurones from the source lobe connected to a single cell in the destination lobe. As a norn is punished then the connections on the source lobe neurones will migrate to other source lobe neurones to form more appropriate 'concepts' for the norn to learn and make decisions from.

Dendrites do not migrate : With this setting dendrites will not migrate at all and stay with the initial connections made at birth. This is most often used when the maximum and minimum number of dendrites is one with a flat spread. Nothing needs to migrate.

Migrate if ANY dendrite is loose and this cell is firing : If a neurone in the current lobe has a number of dendrites attached to a source lobe, this setting will cause the dendrites to migrate if any of these dendrites has a strength of zero and the neurone is currently firing. If a dendrite in the neurone is loose (ie. has strength of zero) but the neurone is not firing then no migration will occur. Only one of the dendrites has to be loose for the migration to occur.

Migrate when ALL dendrites are loose : In this case every dendrite from a given neurone in the current lobe must have a strength of zero for the dendrites to migrate. If only one is zero then no migration will occur.

2.2.5 D0 Dynamics Page

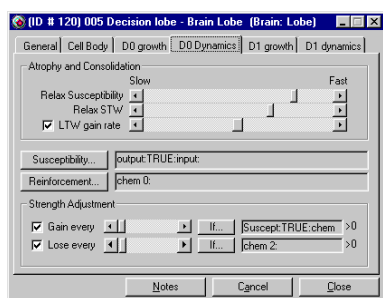


FIGURE 5 – D0 dynamics Interface for Brain Genetics Design

This is probably the most complicated portion of norn genetics. The definition of the dynamics of the dendrites. The 'D0' page describes the dynamics for the 'type0' dendrites. The dynamics defines how the strength value of a dendrite is adjusted, how the STW and LTW are adjusted, etc.

A number of values are associated with a dendrite. A brief description follows to make it easier to understand what the settings on this page mean. Most of these definitions were obtained from the paper by Steve Grand *et al* obtained from the Cyberlife web site.

STW : Short Term Weight - used to modulate input signals. STW is constantly relaxing towards LTW. It is the STW combined with the input value from the source lobe that is used to calculate the 'dendrite value'. The formula used appears to be : $dendritevalue = sourcecell * (stw/255)$ where 'source cell' is the value of the cell that this dendrite is attached to from the source lobe and 'stw' is the dendrites current short term weight value. The formula for calculation of STW appears to be : $stw = ltw + (susceptibility/255) * reinforcement$.

LTW : Long Term Weight - acts as a rest state for STW and provides statistical response to reinforcement. LTW is constantly rising towards STW. LTW and STW move towards each other at different rates (and these rates are defined genetically).

Susceptibility : Current susceptibility to reinforcement. As can be seen in the STW calculation above, the susceptibility modulates the reinforcement value. So the higher the 'susceptibility' value, the more



'reinforcement' affects the short term weight value.

Strength : Controls dendrite migration. Represents how 'strong' the dendrite link is. If strength hits zero then the link is not strong and may migrate.

Atrophy and Consolidation

Relax Susceptibility : This is the half-life of the 'susceptibility' of the dendrite. It is a value from 0 through to 255 and defines a time interval in a similar manner to that of 'Leakage Rate' in 'Cell Body Dynamics'. The value shown in the image above is 20 seconds. So when the susceptibility of a dendrite is set via the Susceptibility SVRule this setting describes how quickly it 'relaxes' back to the rest state. The relationship between this value and the susceptibility SVRule is similar to that of 'Leakage Rate' and 'State SVRule' in the 'Cell Body' page.

Relax STW : This is the half-life or relaxation rate of the Short Term Weight value for the dendrite. The value in the image above represents 5 minutes. Once again it is a number from 0 through to 255 representing a time span. The STW relaxes towards the Long Term Weight (LTW) value of the dendrite. This relax rate sets the speed of that relaxation.

LTW Gain Rate : If this option is checked then LTW constantly rises towards STW at the rate defined in the slider box. This number is a rate in the same manner as all the other leakage rates (even though the genetics kit does not display it as such). For example, if set to 10 seconds (a value of 40) then the LTW will increment by one every 10 seconds until it reaches its rest state (which is the value of STW).

Susceptibility : An SVRule that defines how the susceptibility to reinforcement for that dendrite. The higher the value of the result of this svrule, the more effect 'reinforcement' has on the value of STW. The SVRule defined in the image above basically says if there is an output value for the cell (ie. it is firing) then the susceptibility of the dendrite is equal to that of the input value.

Reinforcement : This is the SVRule used to compute changes in STW. Typically it is set to the value of some chemical. This chemical is adjusted using the various receptors/emitters available in a norn. The STW is increased based upon the value computed by this SVRule modulated by the value of the susceptibility SVrule. In the example above it means that STW is adjusted when the amount of brain lobe chemical 0 changes in this norn.

The relationship between STW, LTW, susceptibility and reinforcement is shown by the following formulae that is used to calculate the values : $dendritevalue = sourcecell * (stw/255)$ and $stw = ltw + (susceptibility/255) * reinforcement$.

Strength Adjustment

Gain Every : If this is checked then the strength value of the dendrite connection will be increased by the value indicated in the slider (from 0 through to 255) if the result of the given SVRule expression is greater than zero. The SVRule given above is 'Suscept :TRUE :chem0 :TRUE :STW'. I believe that this expression means that if the susceptibility of the connection is greater than zero and chemical 0 is greater than zero then the value of the expression is STW. So the dendrite will be strengthened if it is susceptible to reinforcement, has chemical 0 and the STW is greater than zero.

Lose Every : If this is checked then the strength value of the dendrite connection will be decreased by the value indicated in the slider (from 0 through to 255) if the result of the given SVRule expression is greater than zero. The SVRule given above is 'chem 2'. So if the value of chemical 2 is greater than zero then the strength of the dendrite will be reduced.

2.2.6 Dendrite Overview

This is a brief outline of how dendrite values are calculated mostly obtained from the technical papers available at the cyberlife web site. I'll add to this and hopefully provide some programs that demonstrate how things work to make it easier to understand. The following is quoted from one of the technical papers by Steve Grand *et al* :

After disturbance, both the STW and the LTW relax exponentially towards other, with the LTW being the slower. The STW therefore reacts strongly to individual reinforcement episodes, while the LTW effectively computes a moving average of many STW disturbances : if a creature meets with situation X and finds that its chosen course of action was undesirable, then it should immediately be strongly disinclined to repeat the action, especially as many of the incentives to do so may still be present. However, situation X may not always be as bad as first experience suggests, and so the creatures long-term interpretation should be less sweeping



Dendritic Migration. The initial wiring is defined at birth according to a small number of genetic rules. Generally, neurones attempt to connect from one lobe to another in a direct spatial mapping, with multiple dendrites fanning out in a specified distribution to either side of the optimum source cell. After birth, however, individual dendrites may migrate and form new connections (always within the same source lobe). Periodically, a Strength value change is computed for each synapse using SVRules, often in response to chemical changes. If the Strength falls to zero, the dendrite disconnects and follows the appropriate rule about how to find a new connection. These migration rules were chosen in order to fulfill the requirements for the initial brain model.

2.3 Chemical Emitter

2.3.1 Overview

A chemical emitter gene controls the injection of a particular chemical into the norn based upon data retrieved from some Organ, Tissue or Locus within the norn. The Organ, Tissue or Locus is a data value controlled by the Creatures executable ranging in value from 0 through to 255. It can represent a brain cell, external stimuli or other things selected from the dialog box. Every given time period the Organ, Tissue or Locus is examined and compared against a number of settings defined in this gene. Depending upon the results of this comparison an amount of chemical can be injected into the norn.

2.3.2 Dialog Box

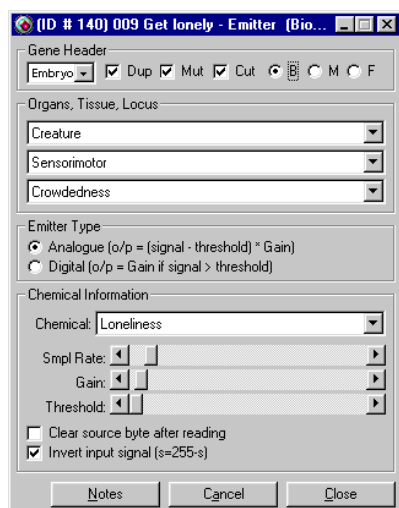


FIGURE 6 – Interface for Chemical Emitter Design

2.3.3 Breakdown – Gene Header

Emitter Type

Analogue : The amount of chemical injected into the norn will be proportional to the signal level received from the locus. The following formula will be applied to know how much chemical to inject : $(\text{signal} - \text{threshold}) * (\text{gain}/255)$.

Digital : A constant amount ('gain') of chemical will be injected into the norn when the 'signal' reaches a specified 'threshold'.

Chemical Information

Chemical : Defines the chemical that will be added to the norn.

Smpl Rate : Sample Rate (inverted) indicates how often the emitter is processed and therefore how often the chemical will be injected.

Gain : Defines how much chemical will be injected. Note that this will increase the amount of chemical into by the given amount, not replace it. See the description of 'Emitter Type' above for the calculation used

Gene Header

Embryo : This field indicates the age at which this gene will switch on. Most genes turn on at the 'Embryo' stage to ensure they operate correctly as soon as the norn is born.

Dup : If this flag is checked then the gene can be duplicated as a genetic mutation. That is, multiple copies of this gene will be able to exist due to mutation.

Mut : Sets whether genetic mutations can be applied to the values contained within this gene.

Cut : If set the gene can be removed during breeding or as the result of mutation.
B/M/F : Defines the gender that this gene will switch on. This is often used to select between genes that involve male or female functionality (eg. hormones).



Threshold : The locus must be firing at this level for a chemical injection to occur. In the case of Digital emitters the signal from the locus must be greater than this threshold. For Analogue emitters the signal is reduced by the amount of this threshold. See 'Emitter Type' above for the calculation.

Clear Source Byte after Reading : After the emitter is processed the locus value will be reset to zero if this option is selected.

Invert Input Signal : The signal value used in all calculations in the emitter will be $255 - \text{signal}$ if this option is selected. This new signal value will then be used for all emitter calculations (including against the threshold).

2.4 Chemical Receptor

2.4.2 Dialog Box

2.4.1 Overview

A chemical receptor gene allows an Organ, Tissue or Locus within the norn to be changed based upon the level of a chemical within that norn. The chemical associated with the receptor is constantly monitored to see if it surpasses a given threshold. When that threshold is reached a formula is calculated from the chemical amount and the result is applied to the Organ, Tissue or Locus selected.

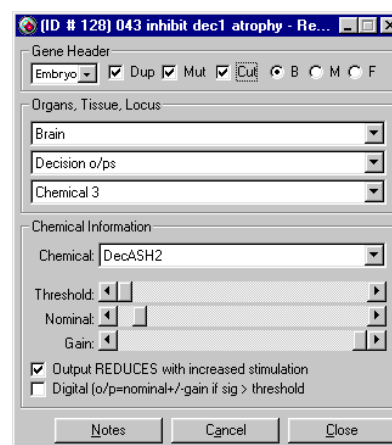


FIGURE 7 – Interface for Chemical Receptor Design

2.4.3 Breakdown

Gene Header : The gene header is the same for all genes.

Chemical Information

Chemical : Defines the particular chemical that will be monitored.

Threshold : Defines the level of the chemical that must exist before the given locus is activated. In the case of Digital receptors the amount of chemical must be greater than this threshold before the locus is activated. For Analogue receptors the signal (ie. amount of chemical) is reduced by the amount of this threshold before calculating the amount to stimulate the locus by. See Formula below for the calculation used.

Nominal : Defines the default or base value used to stimulate the locus. This what the locus will be set to if the amount of chemical is not greater than the threshold.

Gain : Defines what the value of the locus will be set to in the case of a digital receptor or as a scaling factor for calculating the amount in an analog receptor. See Formula below for details.

Output REDUCES with increased stimulation : If this is checked then all adjustments to the base value of the receptor (ie. the Nominal amount) will reduce that base value. If cleared all adjustments will increase the base value.

Digital : The locus will be set to a constant amount if the chemical amount is greater than a certain threshold if this option is checked. If it is unchecked then the receptor is analogue. This means that the locus will be set to a value in proportion to the amount of chemical. See below for details.

Formula

Analogue receptors formula for calculating the value for the locus :

$$\text{Nominal} + (((\text{ChemicalAmount} - \text{Threshold}) * \text{Gain} / 255) * R)$$

Where R is 1 if 'Output Reduces with increased stimulation' is not checked and -1 if it is checked. So the nominal will be reduced or increased based upon this flag.

Digital receptors formula for calculating the value for the locus :

$$\text{Nominal} + ((\text{ChemicalAmount} > \text{Threshold} ? \text{Gain} : 0) * R)$$

So if the chemical amount is greater than the threshold then the locus setting will be the Nominal amount increased or decreased by the Gain depending on the setting of 'R'. If it is not greater than the threshold then the locus setting will be equivalent to Nominal.

2.5 Initial Concentrations

2.5.1 Overview

This gene controls the amount of chemical that will exist within the norn when it is born (or when the gene is activated?). One gene of this kind exists for each chemical within the norn that requires an initial starting value. I don't know the effects of changing the 'switch on' age. If the age is set to something other than embryo does this mean that an amount of chemical will be injected when the creature reaches that age?

2.5.2 Dialog Box

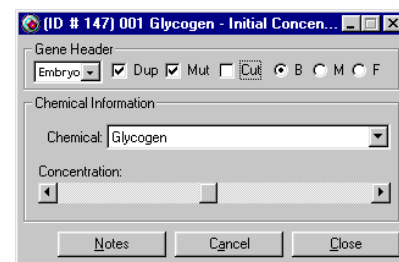


FIGURE 8 – Interface for Initial Concentration Design

2.5.3 Breakdown

Gene Header : The gene header is the same for all genes.

Chemical Information

Chemical : This will be the chemical that will have the initial concentration defined for this gene.

Concentration : The amount of chemical to be initially injected into the norn. It is a value from 0 through to 255.

2.6 Creature Instinct

2.6.1 Overview

An instinct gene provides a way of training the neural net of the norn brain so it will react in certain ways in certain situations. Instinct genes are often used to provide default behavior for things that are very important to the norn. Examples are responding to the verbs typed by the user. These genes do not provide behavior that will always occur. The experiences of the norn during its lifetime can override behavior defined by instincts.

Instincts are processed while a norn is being hatched and while a norn sleeps. It is very important for a norn to get regular sleep so the instincts are constantly reinforced. During this instinct processing time the lobes and neurones define in the gene are set inside the norn brain and the chemical (usually punishment or reward) is processed as if the norn had performed this action.

For example the instinct gene responding to the verb 'come' has the same effect when processed as if the norn had chosen the decision to 'come' when the user typed 'come' and then got a pat from the hand to reward it. This would encourage the norn to do this when awake.



2.6.2 Dialog Box

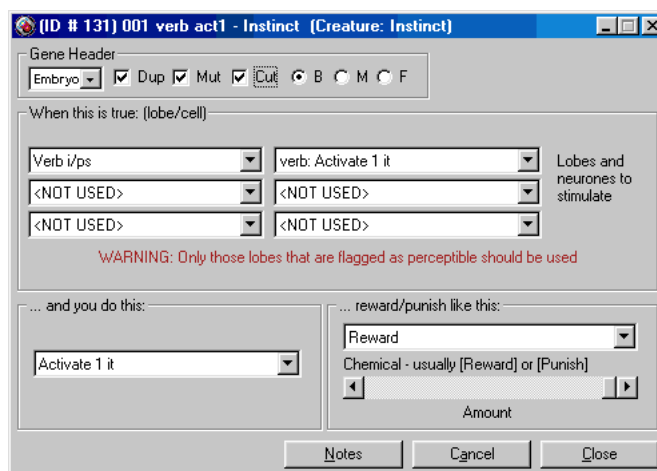


FIGURE 9 – Interface for Instincts Design

2.6.3 Breakdown

Gene Header : The gene header is the same for all genes.

When this is true : (lobe/cell)

Allows selection of the lobes and cells that will be fired within those lobes. This indicates what the norn will need to perceive before it will decide to take the action defined in the 'and you do this' area. A combination of up to three different lobe/cells may be selected. The Genetics Kit gives a warning that only those lobes marked as perceptible should be used. This makes sense as only the perceptible lobes contribute to forming concepts. Any other lobe will cause this instinct to have no real effect.

The lobes and their perceptible settings are :

Lobes and perceptible settings	
Lobe	Perceptible setting
Drive i/ps	Mutually exclusive
Stim source i/ps	No
Verb i/ps	Mutually exclusive
Noun i/ps	No
Gen/sense i/ps	Yes
Decision	No
Attention	Yes

TABLE 1: Lobes and perceptible settings

All lobes with a perceptible setting of 'Yes' or 'Mutually Exclusive' qualify as perceptible lobes so they should be able to be used in instincts.

and you do this...

This is the decision that the instinct will reinforce when the above lobe/cell combinations are active. Think of it as 'if the above lobe combinations occur then the norn will be more or less likely to make this decision depending on the reward/punish settings defined below. *Valid settings are any of the 16 possible decisions or verbs that the norn can perform* :

0. Default (Stay) 1. Activate 1 it (Push) 2. Activate 2 it (Pull) 3. Deactivate it (Stop) 4. Approach it (Come) 5. Retreat from it (Run) 6. Get it (Get) 7. Drop all (Drop)	8. Say what you need (Think) 9. Rest (Sleep) 10. Travel west (Left) 11. Travel east (Right) 12. Decision12 13. Decision13 14. Decision14 15. Decision15
--	--

reward/punish like this

This is where you select the chemical and amount of that chemical that will be injected into the norn when the above settings are made and the instinct processed. Although any of the 256 chemicals can be selected the only two that really make sense are 'Reward' or 'Punish'.

If 'Reward' is selected then the norn will be rewarded and encouraged to perform the action. If 'Punish' is selected then the norn will be discouraged and be less likely to perform the action defined.

2.6.4 Notes

Remember that instincts are only processed when a norn is born and while it sleeps. Too many instinct genes could cause a lot more sleep or longer sleep to be required for all the instincts to be processed. It may pay to have the more important instincts to be the lower numbered genes (as it appears to cycle through in gene number order).

Some instinct genes appear to use the Stim source lobe as an input even though it is not a perceptible lobe. I'm in the process of testing whether these instincts actually work as I suspect they may not. It depends whether the instinct processing mechanism still considers it as perceptible as the Stim source lobe feeds into the Attention lobe which is perceptible. Some research is needed here.

Although only 'Reward' and 'Punish' currently make sense for chemicals, others could be useful in genetically modified norns with different brain/chemical structures. There is scope for some interesting experimentation there. Only three lobe/cell combinations can be defined. This appears to match the number of dendrite links between the perception lobe and the concept lobe in a standard hatchery norn. These norns allow up to three perceptions to form a concept.

2.7 Creature Stimulus

2.7.1 Overview

A stimulus is an event that happens to the norn that it can react to in some manner. The stimulus gene allows the reaction to these stimuli to be defined. This can range from chemicals being injected to cell values in brain lobes being adjusted. There are a number of stimulus responses that are hard coded within the Creatures program. The stimulus genes that are created override these hard coded responses giving a lot of flexibility in determining how the creature will react to the world.

2.7.2 Dialog Box

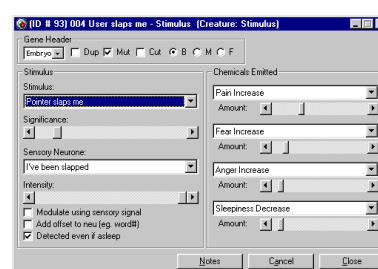


FIGURE 10 – Interface for Stimulus Design



2.7.3 Breakdown

Gene Header : The gene header is the same for all genes.

Stimulus

Stimulus : The stimulus that this gene is attached to. When this stimulus occurs then the effects defined by this gene will be applied. The following stimuli are available :

<ul style="list-style-type: none"> — Disappointment — Pointer pats me — Creature pats me — Pointer slaps me — Creature slaps me — It is approaching — It is retreating — I bump into wall — Object comes into view 	<ul style="list-style-type: none"> — Unrecognised word — Heard user speak — Heard creature speak — I am quiescent (periodic) — I've activated1 — I've activated2 — I've deactivated — I am approaching (periodic) — I have retreated 	<ul style="list-style-type: none"> — I have got — I have dropped — I've stated need — I am resting (periodic) — I am sleeping (periodic) — I am traveling (periodic) — spare action 1-4 — Involuntary action 0-7
---	---	--

Significance : The amount will be increased the neuron relating to the object that caused the stimulus in the Stimulus/Source lobe. This has the effect of making the object appear more interesting and so it is more likely to be looked at. The neuron that gets nudged by this amount appears to differ for each Stimuli (a neuron for each object type in view).

Sensory Neurone : Identifies a cell in the General Sense lobe which will get adjusted by the value given in the 'Intensity' field described below. This allows setting up what the norn can actually sense from the stimuli going on around it. The following cells are available and map directly to cells in the general sense lobe :

<ul style="list-style-type: none"> — <none> — I've been patted — I've been slapped — I've bumped a wall — I am near a wall — I am in a vehicle — User has spoken — Creature has spoken 	<ul style="list-style-type: none"> — Own kind has spoken — Audible event — Visible event — It is approaching — It is retreating — It is near to me — It is active 	<ul style="list-style-type: none"> — It is an object — It is a creature — It is my sibling — It is my parent — It is my child — It is opposite sex — spare 2-13
--	--	--

Intensity : This is the amount by which the particular sensory neurone will be increased by. So if the value here is '50' and the current value of the particular neurone is '25' then when this stimuli occurs the sensory neurone will fire at a value of about '75'.

Modulate using sensory signal : From what I can gather this means that the 'significance' value applied to the stimulus source lobe is first modulated by the value of the neuron associated with the stimulus. I'm not exactly sure what particular value it is modulated by and perhaps it is best explained by describing some of the results I've gotten through experimentation. See the notes section for details. If anyone can provide a clearer explanation I'd appreciate an email.

Detected even if asleep : If the norn is asleep and this option is unchecked then the stimulus will be ignored and none of the effects will be applied. If it is checked then the gene will be processed even if the norn is sleeping.

Chemicals Emitted

Up to four chemicals can be selected along with a particular amount of that chemical to be injected. When the stimulus occurs the defined amount of these chemicals will be injected into the norn.

2.8 Brain Lobes – *Creatures 1*

2.8.1 Overview

The norn brain is divided into a number of lobes. In the standard *Creatures 1* genome there are nine and in the *Creatures 2* genome there are ten.

The brain lobes numbered from 0 through to 8 are hard coded by the *Creatures* executable to perform a particular function. For example, firing a cell in lobe 6 (decision) causes the creature to perform a particular action. The genetic definition of that lobe then defines what happens as a result of this firing. In the case of the decision lobe it is to manage the learning of whether the decision was a good one or a bad one.

The higher numbered brain lobes are not controlled by the *Creatures* executable at all. They are controlled completely through genetics. So for these lobes to do anything a system of receptors, emitters and dendrites must be set up to fire cells or act upon cells firing.

The purpose and function of each lobe is outlined below with descriptions.

Lobe 0 – Perception Copies data from other lobes to form a composite lobe containing all cells that can be used to form concepts.

Lobe 1 – Drive Holds the current values of the creatures drives.

Lobe 2 – Stimulus source Stimulating objects near to the creature cause cells in this lobe to fire so the creature can react to them.

Lobe 3 – Verb Cells in this lobe fire when the user types a verb in the speech box.

Lobe 4 – Noun Cells in this lobe fire when the user types a noun in the speech box or types 'look' with the hand hovering over an object.

Lobe 5 – General Sensory Indicates events that the norn can sense. Usually caused by the effects of processing stimulus genes.

Lobe 6 – Decision An output lobe. By firing a cell in this lobe you cause the creature to perform a particular action.

Lobe 7 – Attention An output lobe. By firing a cell in this lobe you cause the creature to look at a particular object.

Lobe 8 – Concept A storehouse of memories and concepts.

Lobe 9 – Regulator This lobe provides feedback loops for biochemical regulations.

While the lobes have individual functionality it is how they work together that results in the norn brain working. There lobes combine to form a couple of main learning systems that are described briefly in the individual lobe descriptions but will be covered in more detail in future updates :

- Attention seeking system
- Concept learning system
- Decision learning system

Number	Name	X	Y	Width	Height	Neurones
0	Perception	4	13	7	16	112
1	Drive	34	30	8	2	16
2	Source	15	24	8	5	40
3	Verb	37	24	8	2	16
4	Noun	21	3	20	2	40
5	Gen. Sense	32	34	8	4	32
6	Decision	53	15	1	16	16
7	Attention	44	30	5	8	40
8	Concept	12	6	40	16	640

TABLE 2 – Norn Lobes for *Creatures 1*

*Description of lobes positions and size, brain is 64*48. The Regulator lobe, SandraBellum, is not present [12].*



Number	Name	X	Y	Width	Height	Neurones
0	Perception	1	13	7	16	112
1	Drive	34	36	6	3	18
2	Source	15	32	8	5	40
3	Verb	37	32	8	2	16
4	Noun	21	1	20	2	40
5	Gen. Sense	32	40	8	4	32
6	Decision	62	15	1	16	16
7	Attention	44	36	5	8	40
8	Concept	11	5	40	16	640
9	Regulator	4	40	16	1	16

TABLE 3 – Norn Lobes for *Creatures 2* Description of lobes positions and size, brain is 64 * 48 [12].

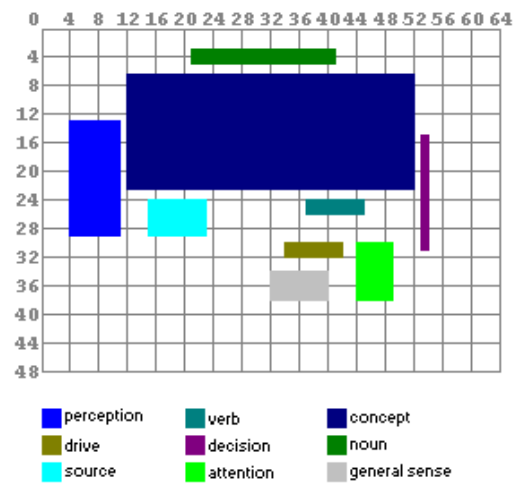


FIGURE 11 – Creatures 1 Brain Map [12]

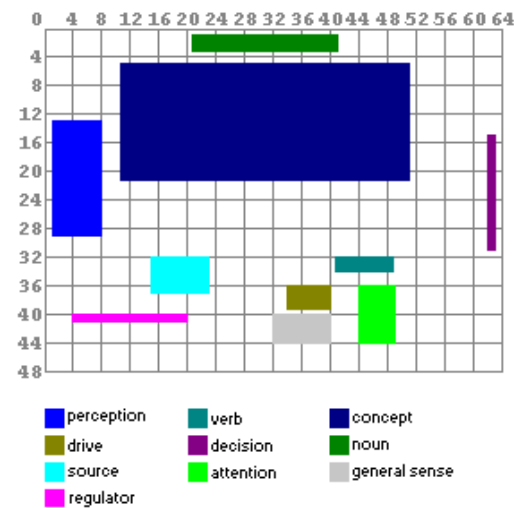


FIGURE 12 – Creatures 2 Brain Map [12]

2.8.2 Lobe 0 – Perception Lobe

The purpose of the perception lobe appears to be to summarise everything that a norn can ‘perceive’ into one brain lobe so the norn can eventually store concepts and memories based upon these perceptions (done in the concept lobe). This tutorial attempts to answer the questions about how the perceptible lobes are summarised into the perception lobe and how to work out what each cell of the perception lobe means.

The usual method of passing cell information from one lobe to another is through wiring up dendrites. Using dendrites can only summarise information from two other brain lobes (the D0 and D1 dendrite connections). With the perception lobe there is more than two lobes that need to be summarised so it appears a mechanism specifically for dealing with the perception lobe was built.

Brain lobes are marked as ‘Yes’, ‘No’ or ‘Mutually Exclusive’ to be copied in the Perception lobe. The brain lobes that are marked as ‘Yes’ or ‘Mutually exclusive’ in an original norn are :

Copied Lobes in Perception		
Lobe	Size	Data copied to perception lobe ?
Drive lobe	16	Mutually exclusive
Verb lobe	16	Mutually exclusive
General sense lobe	32	Yes
Attention lobe	40	Yes

TABLE 4: Copied Lobes in Perception

The perception lobe must have a number of cells equal to or greater than the total number of cells in all lobes marked as 'Yes' or 'Mutually exclusive'. The total number of cells in the above lobes equals 104. The size of the perception lobe is 112 so there is a little room to spare there.

2.8.3 Lobe 1 – Drive Lobe

This lobe holds the current state of the creatures drives (such as hunger, pain, etc). It directly maps to the particular drive chemicals that can be viewed using the science kit. Changing the values of the cells directly in this lobe using CAOS is unlikely to have much of an effect on the norn as the cells are directly set by receptors in the genome. This means that whatever values you manipulate with CAOS commands will be immediately overwritten with the current value of the drive chemical. To change drive values it would seem to be best to change the values of the chemicals either directly or through the drive increaser/reducer chemicals. *The size of the lobe was increased in Creatures 2 to cater for the addition of more drives. The cell values are outlined in the following table :*

Drive Lobe Data Entries <i>Drives</i>		
Cell	Creatures 1	Creatures 2
0	Pain	Pain
1	Need for Pleasure	Need for Pleasure
2	Hunger	Hunger
3	Coldness	Coldness
4	Hotness	Hotness
5	Tiredness	Tiredness
6	Sleepiness	Sleepiness
7	Loneliness	Loneliness
8	Overcrowdedness	Overcrowdedness
9	Fear	Fear
10	Boredom	Boredom
11	Anger	Anger
12	Sexdrive	Sexdrive
13	<i>Not Allocated</i>	Injury
14	<i>Not Allocated</i>	Suffocation
15	<i>Not Allocated</i>	Thirst
16	<i>No such cell</i>	Stress
17	<i>No such cell</i>	<i>Not Allocated</i>

TABLE 5: Drive Lobe Data Entries *Drives*

The drive lobe feeds into the Perception lobe and is marked as mutually exclusive. This means that, at most, one cell from the drive lobe will exist in any given concept that the creature forms. So the creature cannot form the concept of 'hot and hungry'. Only 'hot' or 'hungry' or a combination of these drives with some other perception lobe cell.

2.8.4 Lobe 2 – Stimulus Source Lobe

Cells in this lobe are activated directly by the Creatures executable. The lobe has 40 cells - one for each object classification in Creatures. – Each cell relates to a particular object classification. The cell fires for a given object if that object is within the creatures line of sight and is stimulating in some manner. Stimulus genes can be used to modify this lobe indirectly to indicate whether the object is more or less stimulating. The higher the cell value in a particular cell, the more stimulating that object currently is.

It is a 'winner takes all' lobe, meaning that the highest firing cell will be the only firing cell. This means that the most stimulating object in the area of the norn will be the only object that will appear to be stimulating to the norn. The State value for all the cells is calculated and the cell with the highest state will be the only one with an output value - indicating that it is firing.



The following table outlines the cell values for this lobe :

Source Lobe Data Entries		
Cell	Creatures 1	Creatures 2
0	Current norn	Current norn
1	Hand	Hand
2	Call button	Call Button
3	Water	Nature
4	Plant	Plant
5	Egg	Egg
6	Food	Food
7	Drink	Drink
8	Vendor	Dispenser
9	Music	Implement
10	Animal	Cliff Edge
11	Fire	Detritus
12	Shower	Medicine
13	Toy	Toy
14	Bigtoy	Weather
15	Weed	Badplant
16	Incubator	Nest
17	Blackboard word 33	Badbug
18	Blackboard word 34	Bug
19	Blackboard word 35	Badcritter
20	Blackboard word 36	Critter
21	Blackboard word 37	Seed
22	Blackboard word 38	Leaf
23	Blackboard word 39	Root
24	Blackboard word 40	Flower
25	Blackboard word 41	Fruit
26	Mover	Mover
27	Lift	Lift
28	Computer	Computer
29	Fun	Mediabox
30	Bang	Message
31	Blackboard word 47	Leftright
32	Blackboard word 48	Incubator
33	Blackboard word 49	Teleporter
34	Blackboard word 50	Blackboard word 50
35	Blackboard word 51	Machine
36	Norn	Norn
37	Grendel	Grendel
38	Ettin	Ettin
39	Shee	Shee

TABLE 6: Source Lobe Data Entries

The cell numbers appear to relate to the genus numbers for Objects (COB). So an Object with a genus value of 21 sitting near a norn in Creatures 2 will stimulate cell 21 in the norn, making it think a Seed is nearby.

2.8.5 Lobe 3 – Verb Lobe

The Verb lobe is another lobe that is controlled directly by the Creatures executable. Whenever a verb is entered by the user in the speech box of Creatures then the cell associated with this verb will be fired in this lobe. It's also possible that verbs mentioned by other creatures nearby could fire cells in this lobe but I haven't checked into this yet. – The following table outlines the cell values for this lobe :

Verb Lobe Data Entries		
Cell	Creatures 1	Creatures 2
0	Quiescent	Quiescent
1	Push	Push
2	Pull	Pull
3	Stop	Stop
4	Come	Come
5	Run	Run
6	Get	Get
7	Drop	Drop
8	Think	What
9	Sleep	Sleep
10	Left	Left
11	Right	Right
12	<i>Not Allocated</i>	Eat
13	<i>Not Allocated</i>	Hit
14	<i>Not Allocated</i>	<i>Not Allocated</i>
15	<i>Not Allocated</i>	<i>Not Allocated</i>

TABLE 7: Verb Lobe Data Entries

The word that was learnt for the particular verb slot is what must be typed in for that cell in this lobe to fire. So if the Norn knows 'Push' as 'foo' then typing 'foo' will fire the particular cell. If a complete command is entered like 'alice eat food' then cell 12 in this lobe (for eat) will fire along with the 'food' cell in the noun lobe.

2.8.6 Lobe 4 – Noun Lobe

Cells in the noun lobe are fired when a noun is entered by the user using the speech box of the Creatures executable. It works the same as the verb lobe described above but it is for the objects rather than the action to be performed on the object. It is another lobe controlled directly by the Creatures executable.

The noun lobe will also fire if the user moves the hand over an object and types the command 'look'. This causes the cell for that particular object to fire in the noun lobe. The result of this is the norn usually ends up looking at the requested object. See the description of the Attention lobe for details on how this works.

2.8.7 Lobe 5 – General Sensory Lobe

The cells in this lobe define what the norn can currently sense in the environment. Where the stimulus source lobe is the objects within the environment this lobe is various environmental factors relating to those objects. For example, cells for detecting that the creatures has just been patted, slapped, fallen, whether nearby creatures are the same sex, same species, the parents of the current creature, etc.

The cells in this lobe can be manipulated using the stimulus genes. When a particular stimulus occurs (either by the Creatures executable or using CAOS in a COB) the stimulus gene is activated. That gene can then cause a general sensory lobe cell to fire at a particular intensity so the norn can react or learn from that stimulus.



The lobe is copied to the Perception lobe allowing the cells to be used for forming concepts. This means that a norn can learn whether 'approaching an edge' is good or bad for example.

The following table outlines the cell values for the General Sensory Lobe :

General Senses Lobe Data Entries		
Cell	Creatures 1	Creatures 2
0	I've been patted	I've been patted
1	I've been slapped	I've been slapped
2	I've bumped into a wall	I've bumped into a wall
3	I am near a wall	I am near a wall
4	I am in a vehicle	I am in a vehicle
5	User has spoken	User has spoken
6	Creature has spoken	Creature has spoken
7	Own kind has spoken	Own kind has spoken
8	Audible event	Audible event
9	Visible event	Visible event
10	It is approaching	It is approaching
11	It is retreating	It is retreating
12	It is near me	It is near me
13	It is active	It is active
14	It is an object	It is an object
15	It is a creature	It is a creature
16	It is my sibling	It is my sibling
17	It is my parent	It is my parent
18	It is my child	It is my child
19	It is opposite sex	It is opposite sex
20	<i>Not Allocated</i>	It has pushed me
21	<i>Not Allocated</i>	It has hit me
22	<i>Not Allocated</i>	<i>Not Allocated</i>
23	<i>Not Allocated</i>	<i>Not Allocated</i>
24	<i>Not Allocated</i>	<i>Not Allocated</i>
25	<i>Not Allocated</i>	<i>Not Allocated</i>
26	<i>Not Allocated</i>	<i>Not Allocated</i>
27	<i>Not Allocated</i>	<i>Not Allocated</i>
28	<i>Not Allocated</i>	Approaching an edge
29	<i>Not Allocated</i>	Retreating from an edge
30	<i>Not Allocated</i>	Falling through the air
31	<i>Not Allocated</i>	Hitting the ground post fall

TABLE 8: General Senses Lobe Data Entries

2.8.8 Lobe 6 – Decision Lobe

Each cell in the decision lobe relates to a particular action that the norn can perform. These actions are the same as the verbs in the verb lobe. See the table in that lobe for descriptions of the individual cells.

The Verb lobe is an input lobe - it receives input typed in from the user. The decision lobe is an output lobe. By firing a cell in the decision lobe the Creatures executable will be directed to make the norn perform the particular decision related to the cell being fired. By manually firing cells in this lobe you can force a norn to perform certain decisions. This is how the 'shove' mode of the hand works. When you have the hand in 'shove' mode, pressing a norn results in either the 'left' (cell 10) or 'right' (cell 11) cell to be activated. This will usually cause the norn to move left or right.

The decision lobe is connected to the concept lobe using type 0 and type 1 dendrites. 128 dendrites of each type connect to the concept lobe. Each dendrite link is therefore linked to a particular concept. A concept is a combination of perceptable things (eg. hungry and looking at food, tired, sleepy and just been patted by the hand, etc).

The type 0 dendrites link from decision lobe cells to the 128 concept lobe cells indicate that that particular decision is good if those particular concepts are active. This means that if the concepts become active then the norn will be more likely to choose that decision over other decisions.

The type 1 dendrites link from decision lobe cells to the 128 concept lobe cells indicate that that particular decision is bad if those particular concepts are active. This means that if the concepts become active then the norn will be less likely to choose that decision over other decisions.

2.8.9 Lobe 7 – Attention Lobe

Each cell in the attention lobe relates to a particular object classification in the same manner as the Stimulus Source lobe and the Noun lobe. See the table in those lobe descriptions for the meanings of the individual cells.

The Stimulus Source and Noun lobes are input lobes - they receive input from the environment. The attention lobe is an output lobe. When a cell fires in the attention lobe then the Creatures executable is told to make the norn look at a particular object. If you manually fire a cell in this lobe you will see the 'Creatures View' in the game change to whatever object corresponds to the cell you fired.

Like the stimulus source lobe, it is a 'winner takes all' lobe. This means that only one cell will actually fire - the cell with the highest state value. This makes sense as the norn can only look at one object at a time. The attention lobe copies its information to the perception lobe enabling the current object being looked at to be used in forming concepts.

A mechanism that I call the attention seeking system exists that makes the norn look at the most stimulating object nearby or an object that the user recently asked the norn to look at (by typing 'look' or entering the name of the object). This system works by using dendrite connections from the attention lobe to the stimulus source and noun lobes.

There is one type 0 dendrite linking each cell in the attention lobe to the equivalent cell in the stimulus source lobe. This means that if a cell fires in the stimulus source lobe the value of the firing can be used in state variable calculations in the attention lobe.

There is one type 1 dendrite linking each cell in the attention lobe to the equivalent cell in the noun lobe. Effectively there is a link passing information from the stimulus source and the noun lobe through to the attention lobe.

2.8.10 Lobe 8 – Concept Lobe

By far the largest region of a norn's brain is used to store memories of the events that occur over its lifetime. This region is called Concept Space, and seems to be involved in the collation and organization of a norn's perceptions and experiences. We know that the neurons in this region are very mobile, and are constantly reconnecting themselves to the main sensory lobe in response to new experiences. Artificial stimulation of these neurons can trigger outward behavior, but it is not certain what the relationship is between a given neuron and the behavior it produces. The Concept lobe is the biggest lobe in a Norn's brain. In *Creatures 1* it has a size of 640 neurons.



This lobe receives input signals from the different sense lobes (via the perception lobe in C1 and C2). One neuron of this lobe may have between one and three input dendrites. All inputs of a single neuron are calculated together with a logical AND operation. This is why this lobe is also called "Combination lobe" in Creatures 3 – it simply combines up to three different inputs to one output. Possible meanings may be : "I am hungry AND I see food" or "There is a creature AND it is a Grendel AND it is approaching".

The dendrite's strength is increased by the "reward" chemical and decreased by the "punish" chemical. Dendrites with a strength near zero may migrate to connect to different active input signals, making new random connections possible. The output of the neurons in this lobe is passed to the decision lobe. These dendrites may get positive or negative strength values, so they represent what the creature should or should not do in a certain situation.

2.8.11 Lobe 9 – Regulator Lobe

The regulator lobe (or Sandrabellum named after Sandra Linkletter / slink who created it) is not part of the norn attention, concept, and decision mechanism. It supplies functionality similar to that of the floor receptor/emitters in the receptor and emitter genes but provides a whole lot more flexibility. The following description is almost verbatim from Lis Morris, co-creator of the Canny norns :

It works like the hunger/glycogen neurones used in the life kit, except the whole lobe is dedicated to these kind of positive and negative feedback loops. Therefore, you always see neurones firing in the sandrabellum, since it simply records the amount of some chemicals in the blood stream (norn stream ? they don't really have blood...).

These are then coupled to an emitter that either emits something useful given the level of chemical, or controls future emission of the precursor of that chemical. Here's a list of all the receptor emitter couplings :

Regulator Lobe Loopbacks		
Receptor	Neurone	Emitter
lactate	0	suffocation
This detects drowning, and causes a choke reflex in the norn. Since my alterations to the gene, it should in fact detect myoglobin, IMO.		
Water	1	Thirst
Inverted receptor. nuff said :-)		
Glucose	2	Insulin, hunger
Oops, [name removed - you'll have to guess], two emitters on one locus! I think this is why sometimes norns suddenly collapse when their levels of glycogen, muscle tissue and adipose tissue are low...the emitter suddenly switches to producing hunger, not insulin, essentially knocking the glycogen/glucose equilibrium. I'm going to make a new locus for one of these reactions in my next released genome.		
Adrenaline	3	Stress
Again, pretty self explanatory		
Cholesterol	4	Steroidine
Steroidine is the precursor for cholesterol, and this is a negative feedback loop.		
Testosterone	5	Inhibin
Another negative feedback loop.		
Glycogen	6	Glycogen Synthetase
Negative feedback loop, stops very high levels of glycogen forming		
Bile acid	7	Bilin
Negative feedback, bilin is the precursor of bile acid.		
Adipose tissue	8	Hunger
Thin norns get hungry.		
Starch	9	Fullness
Fat	10	Fullness

Regulator Lobe Loopbacks		
Receptor	Neurone	Emitter
Protein	11	Fullness
These cause that slow reduction in hunger often seen after the initial decrease after eating a fatty food, such as zander fish. I was worried it was production of hunger decrease doing this, and confusing the norms, but obviously not...		
Adipose Tissue	12	Adrenaline
You're too fat, go and exercise ! <G>		
ADP	13	Phosphoglycerokinase
Inverted receptor, causes formation of more glucose for ATP production when ADP is high.		
Muscle Tissue	14	180
<i>snipped note about the history of this one – it's all available on dejanews for those who are curious.</i> Senses high levels of muscle tissue and metabolises them, and stops catabolism of muscle tissue when levels are low.		

TABLE 9: Regulator Lobe Loopbacks

2.8.12 What are Dendrites ?

Dendrites are the links between cells in different brain lobes. They are the means for transferring information from one lobe to another. To help describe what dendrites do I'll use the example of the Attention lobe in a Ron norn.

The Attention lobe has forty cell locations. One for each category of object available in Creatures. The cell with the highest output value in this lobe will indicate the object that the norn is currently looking at (ie. paying attention to). The values for each cell are obtained from other lobes. The two lobes used as inputs to this lobe are the Stim Source lobe and the Noun lobe. Each of these two lobes also have forty cell locations. When a cell fires in one of these input lobes the result is carried down dendrites to the equivalent cell in the Attention lobe. The value of each cells from the two input lobes are summed and the result stored in the corresponding cell of the attention lobe. For example, if Cell 2 of Stim Source fires at value 50, Cell 3 of Stim Source fires at value 20, Cell 3 of Noun fires at value 40 then the result in the Attention lobe will be Cell 2 firing at value 50 and Cell 3 firing at value 50. How do the values of the cells get transferred to the Attention lobe ? By traveling along dendrites. Dendrites are the electrical wiring between lobes.

If you look at the genetics kit dialog box for the Attention lobe you will see 4 tabs related to dendrites. Choose the first one 'D0 growth' and examine the information there. It tells us that the source lobe for these 'D0' (class 0) dendrites is the Stim Source lobe. That is, there is wiring that transfers information from each cell in the Stim Source lobe to cells in the Attention lobe.

The dendrite properties section tells us how many dendrites there are. In this case there is a minimum of '1' and a maximum of '1' spread out using a 'flat' pattern. This means that for each cell in the Stim Source lobe there will be one dendrite connecting it to the equivalent cell in the Attention lobe.

For now we won't worry about the other information. We'll create a norn with two new brain lobes, wire up the two lobes with different dendrite combinations and view the results.

3 Idées générales – «Discussion»

3.1 Définition de règles de bases de conception

3.1.1 Quelques indications générales

On peut codifier le génome d'un individu «Agent» et les informations associées de la façon suivante :

- On retient le principe du fichier unique pour chaque agent (identification nécessaire). Le génome est décrit «ligne par ligne», la liste des variables en tête de fichier, et la «mémoire» de l'agent à la fin (réseau de neurones et autres données).
- Un gène est signalisé par son identification (*header*), et ses paramètres. Un schéma bien défini est utile pour la lecture du génome (début, en-tête, corps, fin) : l'en-tête (*header*) concerne notamment l'utilisation de «drapeaux» pour indiquer les possibilités de mutation / duplication / délétion, et également l'activation du gène, l'âge d'intérêt du gène, la liaison au sexe...
- Une annotation complémentaire au gène. Cette information est signalisée de façon différente du code du gène. La comparaison genes–exons et annotations–introns peut être intéressante.
- Les variables de l'agent peuvent être enregistrées de façon similaire, les informations enregistrées pour le cerveau de l'agent peuvent être enregistrées de cette façon, entraînant une certaine souplesse
- Un séparateur de un ou plusieurs caractères dont l'occurrence est nulle ailleurs dans le fichier : fin de ligne, tabulation...
- La lecture du génome est effectuée au début de la simulation (chargement en mémoire) pour permettre une exécution itérative.

L'utilité du séparateur est d'éviter le chevauchement des informations ou un mauvais fonctionnement du programme. En effet, l'ouverture de fichiers «gen» de *Creatures* montre que la structure génétique des norns est établie selon ce schéma : une succession de *genxxxxxxxxgend* contiguës. Ceci est une reprise et inspiration directe du système utilisé dans *Creatures*. Ce système semble efficace mais peut entraîner un programme long à exécuter lors la lecture du fichier et le chargement d'un individu.

Une idée est d'attribuer à chaque gène une fonction précise, et de permettre un codage qui peut être affecté par des changements. Chaque individu est enregistré avec l'ensemble de son information génétique (IG) : le «décodage» du gène permet son chargement en mémoire. Un ensemble de variables est disponible, notamment lors de l'«exécution» des gènes (sous forme de méthodes informatiques). Un mécanisme pour les mutations, duplications et délétions est à prévoir.

3.1.2 Encodage de l'Information Génétique

aaa	aag	aat	aac	gaa	gag	gat	gac	taa	tag	tat	tac	caa	cag	cat	cac
aga	agg	agt	agc	gga	ggg	ggg	ggc	tga	tgg	tgt	tgc	cga	cgg	cgt	cgc
ata	atg	att	atc	gta	gtg	gtt	gtc	tta	ttg	ttt	ttc	cta	ctg	ctt	ctc
aca	acg	act	acc	gca	gcg	gct	gtc	tca	tcg	tct	tcc	cca	ccg	cct	ccc

TABLE 10 – Tableau des 64 triplets standards du code génétique

Il est nécessaire d'établir comme constituant du «génome» de l'agent un code avec plusieurs caractères, pour l'enregistrement en dehors du programme ou pour tout travail sur les gènes ou les génomes entiers (comparaison, croisement, mutation, duplication, délétion, échanges...). L'idée de base dans ce cadre est l'utilisation du symbole à une lettre des acides aminés) : leur nombre dépend du nombre de commandes distinctes utilisables dans un gène, une même information pouvant être utilisée plusieurs fois. Il faut déterminer les informations possibles et leur code d'appel (passage du «triplet de nucléotides» à l'«Acide Aminé»), selon les critères de sélection : information la plus simple et la plus générale possible, une information unitaire pouvant être facilement utilisable.

Plusieurs étapes de construction du code de l'Information Génétique (IG), associé à une utilisation du dogme central de la biologie (transcription puis traduction $ADN \rightarrow ARN \rightarrow AA$) :

1. Coder gène par gènes suivant le modèle de *Creatures* :
 - Déterminer le code ou la fonction générique pour chaque type de gène.
 - Codage des gènes avec indication du type de gène et des paramètres nécessaires.
 - Établir un fichier génétique exemple avec un gène par ligne.
 - Tester le modèle
2. Passer en code «ADN / ARN / AA» en conservant un gène par ligne (directement codant), un «Acide Aminé» correspond à un caractère unique à transcrire dans l'ordre.
3. Conserver le code de l'étape précédente (toujours un gène par ligne) en ajoutant un système de promoteur et des signaux de fin de gène.
4. *Insérer des introns (parties non codantes, annotations) avec de des signaux «d'épissage» (prévoir éventuellement l'épissage alternatif).*
5. Regrouper les gènes en «chromosomes» (un par ligne) et tester avec du code sans signification entre gènes (idéalement proche des «introns»).
6. «*Petits ARN*» et interférence : créer des gènes qui interviennent au niveau de la traduction ou de l'activité des gènes.

Construction avec les bases “a,c,t,g” (ou autres lettres comme “u,b,v,p”)

Il faut déterminer les gènes de base nécessaires pour un «organisme expérimental» : métabolisme, nutrition, croissance, reproduction... et utilisation des 64 combinaisons pour les triplets, il est possible d'utiliser des quadruplets (256 combinaisons).

Protocole de fabrication de cette fonctionnalité de codage en bases

1. Établir un algorithme pour chaque type gène de manière à ce qu'il soit le plus simple possible mais aussi optimal (nombre de variables, nombre de commandes, rapidité d'exécution...).
2. Obtenir des informations unitaires : caractère, lettre ou chiffre.
3. Transcrire cette information dans un code en triplet ou quadruplet.
4. Observer les résultats à l'utilisation : gène par gène, fonctionnement d'un organisme, conséquences des mutations.
5. Éventuellement : améliorer les algorithmes, créer de nouveaux gènes ou en combiner pour obtenir un comportement global.

Les paramètres nécessaires pour les gènes sont relativement réduits : on peut se contenter d'informations numériques, mais il faut également pouvoir enregistrer les données d'annotation. Les chiffres de 0 à 9 avec des caractères de début et de fin de gène sont suffisants (12 caractères) mais il faut également prévoir des caractères alphabétiques (26 caractères, plus des caractères permettant de séparer les annotations du reste, quitte à utiliser leur répétition).

Ces caractères étant utilisables également pour enregistrer la mémoire des individus ainsi programmés, aucun autre jeu de caractères n'est nécessaire à ce stade. En utilisant une certaine redondance dans le code on peut ainsi compléter et arriver à une bibliothèque de 64 caractères utilisant des triplets, un tel «génome» est facilement lisible par un automate utilisant une table de hashage (structure en arbre de lecture), la longueur du génome importe peu, sauf pour la longueur du chargement en mémoire.

Il est nécessaire de créer des *programmes de tests indépendants* pour la mise en place et le test des gènes (cerveau/lobes avec entrées et sorties ; récepteurs/drivers/emetteurs avec affichage régulier des variables ; ...). Les individus de départ sont pré-établis pour des tests (et éventuellement pour la suite).



Enregistrement des individus

- Enregistrement spécifique au logiciel (sauvegarde en cours).
- À la demande pendant le déroulement de l'application (export)
- Sauvegarde extérieures avec utilisation du format GenBank ?? EMBL ?? FASTA ??

- | |
|--|
| <ol style="list-style-type: none"> 1. <u>Bactéries avec un chromosome comportant 4 groupes de gènes principaux</u> <ol style="list-style-type: none"> (a) Nutrition (recherche d'énergie : critères nutritifs des éléments fixés au départ) (b) Assimilation (transformer l'énergie en vue d'une utilisation ou d'un stockage éventuel) (c) Croissance (développement, mouvement...) (d) Reproduction (et mutations variées plus ou moins rares, croisements si reproduction sexuée ou échanges de gènes : ponts cellulaires et virus) 2. Mort par sureffectifs, disparition des moins adaptés... 3. Mutation est substitution, deletion ou addition d'une base (si codage en bases). 4. Lecture de l'ADN : de M (méthionine) à * (stop) ou une suite de caractères identiques. |
|--|

TABLE 11 – Expérimentation de base avec une bactérie de test.

Rien ne doit être modifié dans le programme au cours de son exécution par un utilisateur : toute demande sauvegarde se fait en externe (à moins d'une fonctionnalité précise de ré-initialisation), il est utile dans ce cadre de disposer d'un outil d'import / export dans la simulation.

Un tel outil (intégré ou non au logiciel) peut permettre l'analyse et la manipulation des génomes des individus (agents, organismes) intégrés au modèle. De plus, l'idée est de pouvoir créer de nouveaux individus avec un outil facile à manipuler et permettant des croisements ou une observation directe des individus.

3.1.3 Utilisation du «multithreading»

Ceci est prévu pour la fin mais ne doit pas être oublié lors du développement du logiciel ; le choix a porté sur une modélisation synchrone plutôt que asynchrone, associé à une gestion de la priorité des processus (*Thread*).

- alléger le programme avec un cycle d'exécution itérable,
- utilisation de fonctions indépendantes avec conservation et modification en mémoire vive.

3.2 Idées d'intégration dans le modèle**3.2.1 Agents infectieux : bactéries, virus et plasmides**

Le principe est ici de définir des parasites pouvant s'auto-répliquer et s'auto-modifier dans un organisme / individu / agent hôte et entraînant éventuellement des perturbations dans celui-ci. Un tel parasite entraîne la production d'Antigènes (Ag) nécessaires pour les production de synthèses bactériennes ou virales ; ceci est contré par des Anticorps (Ac) de l'hôte. Il y a insertion d'un code parasite dans le génome hôte, ce qui entraîne une biochimie particulière, voire des comportements particuliers (production de particules virales par exemple).

Propriétés des plasmides bactériens

Réplication autonome (dans l'hôte).	Incompatibilité des plasmides apparentés.
Fertilité (autotransfert d'un hôte à l'autre).	Mobilisation (utilise hôte pour transmission).
Modification (gènes en + ou en -).	Production de toxines.
Fusion des réplicons.	Résistance/sensibilité (à certains facteurs).
Mutagène/antimutagène.	Caractères métaboliques.
Facteurs adhérence/pathogénicité.	Recombinaison/transposition des gènes.
Capacité d'induction de tumeurs.	

Ces éléments génétiques ne font strictement pas partie du génome de l'agent (un chromosome «à part» supplémentaire) mais peuvent éventuellement y être intégrés de façon définitive (dans la modélisation) comme des gènes d'agents infectieux (bactéries, virus ou plasmides).

- Informations pour le transfert avec plus ou moins d'autres gènes.
- Transmission du donneur (F^+) au receveur (F^-), ce dernier ne possédant pas le «plasmide», qui peut être perdu ou éliminé.
- Transmet aussi lors de la scissiparité des individus (reproduction asexuée).
- Influence sur la sélection des individus (caractères métaboliques, sensibilité / résistance).
- HFr = comme les F^+ , mais au lieu du simple plasmide, c'est l'information génétique complète qui est copiée chez le receveur (copie totale ou fragmentaire, le receveur ne devient pas forcément donneur).

3.2.2 Système immunitaire

Le système immunitaire chez les organismes évolués utilise le système des Anticorps (Ac) composés de parties fixes de parties variables (code de reconnaissance spécifiques) ; spécificité de réaction avec des Antigènes (Ag) d'origine extérieure. Le Système Immunitaire (SI) est un programme de reconnaissance général qui agit si un Ag n'est pas dans la base de l'individu, est inconnu, ou reconnu comme «étranger» (déjà rencontré auparavant).

De façon générale dans le modèle créé ici, il s'agit plus simplement d'utiliser certaines variables comme nécessaires au développement des agents infectieux et déclencheurs des comportements infectieux (Antigènes qui provoquent la production de nouveaux objets comme des particules virales). Pour contrer ces réactions, l'organisme hôte possède des réactions biochimiques (associées à des variables Anticorps) qui font diminuer la quantité d'antigènes présents afin d'éviter les réactions liées au métabolisme importé par le parasite.

3.2.3 Étude évolutive

L'idée est d'effectuer une étude évolutive au sein d'une telle simulation et de travailler à partir d'une taxonomie (classification selon Phylum, Classe, Ordre, Famille, Genre, Espèce, Variété). L'étude s'effectuant au départ par phylum et par espèces directement modélisées. Une définition des phylum de départ est la suivante :

- *Silicobacter* : «bactéries»
- *Silicoviridae* : «virus»
- *Silicoviriditae* : «plantes»
- *Silicoanimae* : «animaux»

Un cinquième phylum *Silicodaemon* est prévu pour définir les agents programmés dans l'environnement mais non susceptible d'évolution, pour assurer un renouvellement de l'environnement. L'objectif est de définir des espèces de départ dans les cinq phylum et étudier l'évolution obtenue.

Les outils classiques de comparaison de séquence seront ensuite utilisés pour classer les espèces et séquences obtenues suite à l'évolution (dot-plot, algorithmes SW et NW, phylogénie, FASTA, BLAST...).

3.2.4 Gènes particuliers

Une idée concerne l'utilisation de gènes «spéciaux» utilisant les définitions pré-établies mais amenant des comportements particuliers et ayant un impact au niveau évolutif (autres que ceux des agents infectieux). Quelques idées glanées ici [3, 4] et là [23] à propos de gènes théoriques connus pour leur «performance», enregistrés, encodés dans les génomes et plus ou moins activés (selon stress ou autre activité particulière). Il peut s'agir de gènes naturellement présents dans les génomes ou de gènes viraux plus ou moins actifs provoquant des comportements particuliers ou des remaniements de génome.

- Des **éléments transposables** [27, 32, 33] : transposons, rétrotransposons et éléments viraux analogues... Ces éléments peuvent amener à un remaniement du génome «hôte».
- **Req Queen (RQ)** : engendre des phénomènes de Mutation / Duplication / Délétion des gènes lors de la vie d'un individu et non plus seulement lors de la reproduction, remanie l'ordre des gènes, voire provoque la fusion entre deux individus de la même espèce ou d'espèces différentes.
- **Chi (A/B/C)** [23] : marqueur d'une capacité particulière, qui provoque un comportement inné (réseau de neurone ou lobe figé).
- **SHEVA** [3, 4] : comme les gènes RQ mais SHEVA provoque l'apparition d'un virus qui amène le phénomène à s'exprimer chez d'autres individus, via une transmission virale et / ou des phéromones.



3.2.5 Reproduction asexuée, sexuée...

On peut distinguer deux grands modes de reproduction, sexuée et asexuée. L'idée est de reprendre le principe biologique de reproduction : le cycle de vie cellulaire (pour la reproduction asexuée) et de préparation hormonale (seuil de fertilité, cycle de fertilité).

— Reproduction asexuée

Copie du génome et apparition d'un organisme identiques («scissiparité») à quelques modifications près (mutation, duplication, délétion) avec remise à zéro du niveau hormonal si il y a utilisation d'un cycle hormonal (une ou plusieurs hormones).

— Reproduction sexuée

Au moins deux sexes, *parthénogénèse possible*, **fabrication de gamètes lors de l'atteinte de seuil(s) hormonaux (stimulus)** :

— **mâle** avec seuil hormonal déclenchant la fertilité («testotérone»).

— **femelle**, cycle hormonal («œstrogène», «progestérone»...).

— autres sexes, plutôt hypothèse d'un système de cycle.

Si il y a contact de gamètes ou lors de phases communes de fertilité, ceci provoque la réalisation du croisement entre les gamètes de différents individus de sexes différents et un changement de production des hormones cycliques (pour l'individu porteur / producteur de l'œuf. *Si le nombre de gènes ou la taille du génome est significativement différent(e), plusieurs possibilités sont admises* :

— arrêt car espèces différentes (on considère une espèce possède un nombre de gènes défini avec une tolérance) ;

— fusion des gamètes (et création d'une «nouvelle espèce»).

Le croisement des génomes des parents s'effectue via l'association des gamètes créés (choix aléatoire des gènes, crossing-over si polyploïdie) quelques modifications peuvent intervenir (mutation, duplication, délétion).

— Les particularités du nouvel individu sont liées à son patrimoine génétique, à ce niveau, il y a nécessité de créer un **identifiant unique** qui peut être codé génétiquement...

Un questionnement à ce niveau est la conception d'espèces **ovipares ou vivipares**, notamment pour les espèces sexuées. Il est sûrement plus simple de «provoquer» la création d'un œuf et de permettre son évolution en individu à différents stades (œuf, embryon / larve...) ; il suffit de prévoir prévoir un emplacement de stockage des gamètes dans les individus (tous sexes confondus) et éventuellement pour le croisement des gamètes.

Une autre questionnement concerne la ploïdie des organismes modélisés, au départ modéliser des organismes haploïdes posera moins de problèmes pour la reproduction tant asexuée que sexuée (copie du génome, crossing-over). La complexité de l'utilisation de la polyploïdie intervient dans la reproduction sexuée (nécessité des gamètes), la parthénogénèse peut être utile à ce stade pour tester une reproduction sans croisement.

Le **but des modifications telles que mutations / duplications / délétions** est d'éviter l'uniformisation de l'Information Génétique (IG) et s'effectue sur chacun des gènes (ou sur l'ensemble du génome avec des indels). Il se pose également la question de l'activation / inactivation des gènes dans le génome d'un individu d'où l'intérêt de l'en-tête dans l'encodage des gènes.

3.2.6 Parthénogénèse – *Extraits de Wikipedia* [41]

La parthénogénèse (ou parthénogénèse) est la multiplication à partir d'un gamète femelle non fécondé. Ce phénomène s'observe naturellement chez certaines espèces végétales et animales, mais peut également être provoqué artificiellement. La parthénogénèse est une reproduction monoparentale. Cette reproduction a un avantage sélectif car elle produit un grand nombre d'individus sans la présence de l'organisme mâle. Ce phénomène donne :

- parthénogénèse thélytoque : uniquement des femelles,
- parthénogénèse arrhénotoque : uniquement des mâles,
- parthénogénèse deutérotoque / amphotérotoque : mâles et femelles.

Selon les hypothèses actuelles la multiplication asexuée peut avoir des avantages à court terme quand la croissance démographique est rapide ou quand l'environnement est stable. Au contraire la reproduction offre à long terme un net avantage en permettant une adaptation plus rapide à des environnements changeants. Les lignées se reproduisant par multiplication asexuée peuvent accroître leur nombre rapidement parce que, comme les individus sont toujours femelles, chacun peut produire des œufs qui éclosent. Dans les populations séparées en sexes certains individus sont mâles et ne peuvent donc pas avoir de progéniture. Autant dire que dans des conditions idéales une lignée asexuée aura en gros un taux de croissance démographique double par rapport à une population composée pour moitié de mâles, c'est ce qu'on appelle le désavantage reproductif et plus couramment le «two-fold cost of sex». Les organismes qui peuvent se reproduire par parthénogenèse sont aussi plus capables de coloniser des habitats isolés comme les îles océaniques, puisqu'il suffit qu'un seul exemplaire de l'espèce (nécessairement femelle) atteigne l'habitat pour commencer à le peupler.

Une autre conséquence de la reproduction asexuée, qui peut avoir autant d'avantages que d'inconvénients, c'est que la progéniture est génétiquement identique ou presque identique à son parent (sauf en cas de mutation). Cette similarité génétique peut être favorable si le génotype convient exactement à un environnement stable, mais elle est désavantageuse si l'environnement change. Par exemple, s'il apparaît un nouveau prédateur, un nouvel environnement, ou un nouvel agent pathogène auquel un individu est mal adapté, sa lignée parthénogénétique sera tout aussi vulnérable que lui. Par contre, une lignée produite par reproduction sexuée a de meilleures possibilités d'adaptation grâce à la recombinaison génétique par laquelle chaque individu présente un génotype original.

3.2.7 Biochimie et voies métaboliques

Une idée majeure de la construction de ce système de modélisation est d'intégrer un système biochimique aux organismes modélisés, avec un ensemble de données sur la chimie des agents, ainsi que des gènes codant l'utilisation de ces éléments chimiques. L'ensemble des éléments chimiques utilisés dans le modèle sont simplement rassemblés dans une même table, avec un ensemble de règles d'utilisation.

De façon complémentaire aux éléments simples (liste issue de la table de Mendeleïev), ce tableau est prévu pour comprendre également les composés chimiques (molécules, composés de molécules...) ainsi que différents «drapeaux» utiles pour modéliser un organisme. L'objectif est de reprendre le modèle connu de différentes voies métaboliques des organismes procaryotes et eucaryotes, dans leurs différentes étapes : glycolyse, bêta-oxydation des acides gras (hélice de Lynen), cycle de l'acide citrique (Cycle de Krebs)... Chaque enzyme impliquée (catalysant une réaction chimique) étant reprise dans un modèle biochimique encodé génétiquement dans le modèle, dans chacun des organismes.

Métabolisme des sucres – *Glycolyse*.

1. Activation des hexoses (phosphatation du glucose).
2. Isomérisation (Fructose 1,6 phosphate).
3. Trioses phosphates.
4. Récupération de l'énergie (glycerate, pyruvate...).

Métabolisme des acides gras

Hélice de Lynen

1. Activation de l'Acide Gras (Acyl-CoA à n carbones).
2. Déshydrogénation Acyl-CoA.
3. Hydratation double liaison.
4. Déshydrogénation.
5. Coupure (acétylCoA + AcylCoA à $n-2$ carbones).
6. Retour autant que nécessaire au deuxième point.

Métabolisme énergétique

Cycle de Krebs

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.



Nombre de carbones	Nom usuel	Nom IUPAC	Nomenclature physiologique	Formule chimique semi-développée
8	acide caprylique	acide octanoïque	C8 :0	H3C-(CH2)6-COOH
9	acide pèlargonique	acide nonanoïque	C9 :0	H3C-(CH2)7-COOH
10	acide caprique	acide décanoïque	C10 :0	H3C-(CH2)8-COOH
11	acide undécylique	acide undécanoïque	C11 :0	H3C-(CH2)9-COOH
12	acide laurique	acide dodécanoïque	C12 :0	H3C-(CH2)10-COOH
13	acide tridécylique	acide tridécanoïque	C13 :0	H3C-(CH2)11-COOH
14	acide myristique	acide tétradécanoïque	C14 :0	H3C-(CH2)12-COOH
15	acide pentadécylique	acide pentadécanoïque	C15 :0	H3C-(CH2)13-COOH
16	acide palmitique	acide hexadécanoïque	C16 :0	H3C-(CH2)14-COOH
17	acide margarique	acide heptadécanoïque	C17 :0	H3C-(CH2)15-COOH
18	acide stéarique	acide octodécanoïque	C18 :0	H3C-(CH2)16-COOH
19	acide nonadécylique	acide nonadécanoïque	C19 :0	H3C-(CH2)17-COOH
20	acide arachidique	acide eicosanoïque	C20 :0	H3C-(CH2)18-COOH
21	-	acide hénéicosanoïque	C21 :0	H3C-(CH2)19-COOH
22	acide béhénique	acide docosanoïque	C22 :0	H3C-(CH2)20-COOH
23	-	acide tricosanoïque	C23 :0	H3C-(CH2)21-COOH
24	acide lignocérique	acide tétracosanoïque	C24 :0	H3C-(CH2)22-COOH
25	-	acide pentacosanoïque	C25 :0	H3C-(CH2)23-COOH
26	acide cérotique	acide hexacosanoïque	C26 :0	H3C-(CH2)24-COOH
27	-	acide heptacosanoïque	C27 :0	H3C-(CH2)25-COOH
28	acide montanique	acide octacosanoïque	C28 :0	H3C-(CH2)26-COOH
29	-	acide nonacosanoïque	C29 :0	H3C-(CH2)27-COOH
30	acide mélissique	acide triacontanoïque	C30 :0	H3C-(CH2)28-COOH
31	-	acide hentriacontanoïque	C31 :0	H3C-(CH2)29-COOH
32	acide lacéroïque	acide dotriacontanoïque	C32 :0	H3C-(CH2)30-COOH

TABLE 12: Nomenclatures des acides gras saturés linéaires de 1 à 32 carbones

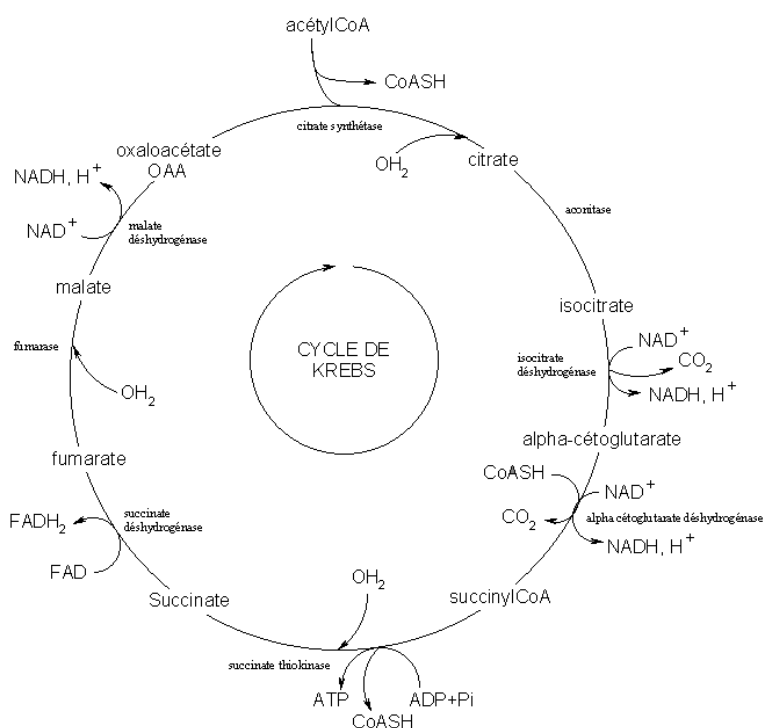


FIGURE 15 – Schéma de déroulement du cycle de Krebs.

3.3 Mise en place de la programmation

3.3.1 Représentation objet

- Discretisation de l'espace avec des variables définissant des espèces chimiques suivant une nomenclature pré-établie (tableau d'ensemble des atomes et molécules).
- Définition générique des agents, et établissement éventuel de catégories (espèces vivantes et familles, éléments d'interaction...) avec pour les éléments vivants ou dérivés (animaux, plantes, fruits...) une définition d'espèces chimiques ainsi qu'un génome.
- Typage fort des gènes (classes de gènes, paramètres, méthode d'exécution).
- Interactions normalisées avec l'environnement (méthodes et fonctions de transfert).

3.3.2 «Chimie» – métabolisme et génétique

- *Chemicals*
Ceci désigne les variables d'un objet et concerne (avec les gènes) les agents actifs comme un organisme ou un élément qui en est issu (fruit par exemple).
- *Initial Concentration*
Ce type de gène existe pour initialiser les variables chimiques d'un individu (agent, organisme, œuf).
- *Biochemical Reaction*
Les gènes de ce type permettent les réactions chimiques d'un organisme.
- *Emitter* et *Receptor*
Les gènes de ce type permettent l'émission et la réception d'éléments chimiques (phéromones).
- *Stimulus, Decision, Brain, Lobe, Instincts...*
Il s'agit de gènes d'interactions avec l'environnement (*Stimulus, Decision*) et de construction de réseaux neuronaux d'apprentissage (*Brain, Lobe*), dont certains sont pré-établis (*Instincts*).

3.3.3 Interactions entre l'individu et son environnement

cL'idée d'un système nerveux, minimaliste ou plus élaboré, est intéressante à plusieurs titres, notamment pour la mise en place d'un système de détection (stimulus, sens de perception) ou d'attention et d'action sur des objets.

- *Emetteurs / récepteurs internes et externes*
Une variation des gènes d'émission et de réception est possible pour interagir de façon «chimique» avec l'environnement, de la même façon que des gènes internes le font avec un système de gestion de la biochimie interne (un système nerveux). Un exemple de ce type de gène d'action externe est l'émission et la détection de phéromones, pour les variants internes il s'agit de boucles d'interactions (variations chimiques plus élaborées que les réactions biochimiques).
- *Stimuli et décision*
Détection d'éléments extérieurs à l'individu (agents dans le même *Container*) et stimulation à partir de ces éléments ou de décision d'action sur ces éléments (récupérer, déposer, manger, déplacement...); ces gènes interviennent également sur le système nerveux.
- *Instincts*
Ce type de gènes a pour objectif de créer des connections entre neurones à partir d'un schéma pré-établi et codé génétiquement, afin de permettre une réaction directe du système nerveux dès le début (prise de décision suite à un stimulus). Ceci n'empêche pas l'apprentissage de nouveaux schémas lors de la vie de l'individu, ce qui peut servir pour constater l'effet Baldwin [1].
- *Cerveau et lobes*
L'intérêt de ces types de gène est de réserver un espace à un réseau de neurones et aux règles qui régissent les neurones (de façon groupée) afin de créer un ensemble homogène au niveau implémentation, mais potentiellement très hétérogène dans ses fonctionnalités.

3.3.4 Types d'agents : automates, plantes, bactéries, fourmis...

- **Automates** : métabolisme, itérations, mémoire
- **autoreproducteurs** (copie à la mutation près)
- **croiseur/sexué** : combine son code avec celui d'un autre (gamètes, œufs, copie partielle du génome)
- **Plantes** : produisent des fruits, si les fruits ne sont pas consommés, après un certain délai, une nouvelle plante apparaît (dans un premier temps, les plantes ne produisent que des fruits non reproductifs).
- **Fourmilière** : reine, œufs, ouvriers, guerriers, reproducteurs ; évolution des individus ou pré-déterminés ? Phéromones (nourriture, contrôle reproductif...).
 - **Reine** : ID de la colonie, possède ensemble du génome, produits œufs « partiels » selon besoins (nourriture, défense, attaque, colonisation...),
 - **Œuf** créé avec une partie déterminée du génome : donne individu doté de fonctions précises et doté d'une vie plus courte.
 - **Ouvrier** : recherche objet (œuf, nourriture), transport et rassemblement des objets près de la reine (localisation en mémoire), laisse des marques (même ID que la reine).
 - **Guerrier** : demande ID, détruit agents ayant ID différent, laisse des marques (même ID que la reine).
 - **Reproducteurs** : voyageur longue distance, mâle ou femelle (selon moitié du génome), s'apparie avec un autre pour donner code entier de nouvelle reine (nouveau ID, mutations, erreurs de coupage de génome, duplication ou délétion de gène...).
- **Réseau de fourmillières / mites** : sur le même principe que la fourmilière précédente, mais les œufs produits donnent des individus qui passent par tous les stades (et finissent par créer de nouvelles colonies).

3.3.5 Codes de transcription, traduction...

De la même façon que le code génétique de traduction entre les acides nucléiques et les protéines (un code génétique standard et ses variantes selon les Phylum, et Divisions), il est intéressant à plusieurs titres d'encoder « génétiquement » l'exécution des diverses fonctionnalités des Agents (surtout les Organismes). Le premier intérêt est que cela permet de définir des fonctions génériques pour chacun des types de gènes prédéfinis, auquel cas seuls le type de gène et les paramètres correspondants sont à encoder (codage paramétrique), la fonction d'un gène étant directement exécutable. Un autre intérêt est de transcrire le code à exécuter, qui est dynamiquement traduit et évalué, voire compilé et exécuté si ce code est valable (encodage complet).

De la même façon que le code génétique en biologie est redondant, un tel codage informatique peut permettre de réduire les effets des mutations, mais un intérêt est d'étudier l'impact des mutations et de la sélection (tri global et notation par rang des individus de façon aveugle, selon la traduction et l'exécution de leur génome). Une telle comparaison entre un génome biologique et un code de programme informatique permettrait de faire évoluer un code informatique de la même façon qu'une espèce évolue, non pas dans une recherche de perfection, mais d'optimum dans un environnement donné : les outils et méthodes de comparaison et de classification (homologie, phylogénie...) peuvent se révéler utiles ici à plusieurs titres pour analyser cette évolution : FASTA, BLAST, UPGMA, Neighbor-Joining...

Un tel modèle de développement de simulation, proche des algorithmes génétiques, permettrait ainsi d'évaluer non seulement des modèles et simulations biologiques, mais aussi des modèles informatiques de programmation (algorithmes, analyses, traitement et échanges de données...) dans des contextes donnés, selon l'environnement créé (système d'exploitation, réseau, tests de compatibilités entre programmes) complémentaire des outils existants.



4 Développement

4.1 Variables, types de gènes... Notes d'implémentation.

4.1.1 *Chemicals* – Variables

Le choix a porté sur l'intégration d'un ensemble de 1000 variables entières, numérotées de 000 à 999, accessibles facilement, pour les gènes amenés à consulter ou à modifier ces variables. Les valeurs de ces variables sont initialisées à 0 et sont prévues pour une valeur maximale de 999, ceci dans un soucis de future compatibilité lors des enregistrements et de rechargement.

4.1.2 *GeneHeader* – En-tête

Un type générique de gènes a été implémenté, dont tous les autres gènes vont hériter, et qui comporte les éléments d'en-tête pour tous les types de gènes. Cet en-tête comporte des indications sur les possibilités du gène à muter, être dupliqué ou à être délété. De plus, des indicateurs fournissent également des informations sur l'activité du gène, l'âge minimal et l'âge maximal d'activation du gène, ainsi que le sexe d'activation (avec une valeur par défaut pour activer le gène quelque soit le sexe de l'individu).

4.1.3 *InitialConcentration* – Concentrations initiales

Le développement de ce type de gènes est simple, seuls deux paramètres sont nécessaires : la variable à modifier et la valeur qui doit lui être attribuée. Le soucis de compatibilité est le même que pour les variables, les paramètres de ce type de gène vont de 000 à 999.

4.1.4 *BiochemicalReaction* – Réactions Biochimiques

Pour ce type de gènes, neuf variables sont nécessaires pour indiquer l'ensemble de la réaction biochimique codée par ce gène. Les paramètres (dont les valeurs vont de 000 à 999) sont indiquées avec l'équation de la figure 16. Les réactions possibles sont celles du gène correspondant dans *Creatures* et décrites à la page 6.

$$a * A + b * B \longrightarrow c * C + d * D; \text{ Vitesse de réaction : KM}$$

[a-d] \rightarrow [A-D]coef : Coefficient pour le réactif [A-D]

[A-D] \rightarrow [A-D]chem : Numéro de variable du réactif [A-D]

KM \rightarrow KM_Vmax : Coefficient de vitesse

FIGURE 16 – Formule descriptive du gène *BiochemicalReaction*

[a-d] indique a, b, c ou d avec respectivement [A-D] indiquant A, B, C ou D.

4.1.5 *BrainGene* – Cerveau

Le seul intérêt de ce type gène est de décrire la hauteur (*height*), la largeur (*width*) et la profondeur (*depth*) réservés aux neurones du système nerveux. Seules la hauteur et la largeur sont pour l'instant utilisées. Tous ces paramètres sont compris entre 00 et 99. Ce type de gène implique une réservation d'espace mémoire (et de prévision de temps de calcul lié au réseau de neurones) qui est le principal facteur limitant de la simulation, lié aux calculs nécessaires pour les états neuronaux

4.1.6 *BrainLobeGene* – Lobes du cerveau

Ce type de gène décrit la position (coordonnées x et y), la hauteur et la largeur des lobes au sein du cerveau (si celui-ci est préalablement défini avec un gène correspondant) et ont des valeurs entre 000 et 999. D'autres paramètres sont également dans ce type de gène et concernent les propriétés des neurones contenus dans ce lobe, le niveau de repos, le niveau d'activation, la valeur de diminution, le nombre minimal et le nombre maximal de dendrite, la proximité des connexions établies, la possibilité de reproduction des neurones et l'indication si le lobe contient un seul neurone actif ou non (comportement WTA).

Les neurones sont totalement configurables dans leur comportement au sein des lobes de façon isolée ou globalement (lobes d'un seul neurone, comportement WTA du lobe, nombre de connexions...). L'intérêt est de pouvoir grouper les neurones selon l'intérêt de leur activité et de la gestion des entrées et des sorties du cerveau, associés à des gènes émetteurs-récepteurs ou de stimulus.

4.1.7 *EmitterReceptor* – Émetteurs et récepteurs

L'intérêt de ce type de gène est l'interaction des variables chimiques avec le système nerveux d'un organisme donné, soit pour indiquer l'atteinte d'un seuil chimique (récepteur) soit pour émettre une espèce chimique à la demande (émetteur). Ceci s'effectue prioritairement sur les variables chimiques internes à l'organisme ; ensuite l'intérêt des phéromones sur leur dépôt et leur détection a amené à élargir leur effet aux variables de l'environnement courant (émetteurs et récepteurs externes dans ce cas). Les variables de ce type de gène sont le numéro de la variable concernée, le niveau d'activation (du neurone ou de la variable chimique), la valeur à ajouter (à la variable ou à l'état du neurone), la position du neurone, un indicateur «émetteur ou récepteur» et un indicateur pour en connaître l'internalité.

Les émetteurs et les récepteurs peuvent être amenés à avoir un rôle mineur dans le fonctionnement du système nerveux, en-dehors des signaux liés à la présence de phéromones (contact avec les variables extérieures locales). Une application interne du système nerveux est l'intervention de boucles de rétro-action (*loop-back*) dans l'organisme concerné pour provoquer certaines réactions chimiques ou certains comportements complexes.

4.1.8 *StimulusDecision* – Perception et décision

Ce type de gène est proche du type *EmitterReceptor* sur le principe de base : il s'agit de recevoir des données et de les intégrer aux variables ou d'accomplir des actions en fonctions des variables. Les paramètres de ce type de gène sont : l'indicateur d'entrée ou de sortie (perception ou décision), un indicateur de variable ou d'objet (liste pré-établie), le numéro de variable ou d'objet source (de 000 à 999), le seuil d'activation, un attribut (direction, type d'objet, état), la variable destination, la valeur à ajouter dans cette variable et l'action à effectuer.

Un parcours global de ce type de gène est nécessairement relié à une mémoire (une partie des variables), pour la partie de *prise en compte des décisions* dont la prise en compte est découplée de la génétique, mais aussi pour la réception des stimulus. Cette mémoire indique les actions à accomplir avec éventuellement certains paramètres associés (direction, objet) et est décrite page 44.

L'intérêt d'un typage fort et d'une nomenclature des objets et des directions dans l'environnement est nécessaire à ce niveau, ceci permet notamment la détection de phéromone dans des cases voisines. Ce type de gène n'est pas obligatoirement lié à un système nerveux, mais peut l'être via les gènes *EmitterReceptor*. Dans ce cas, un lobe d'attention et de décision utilise la fonctionnalité WTA des lobes pour assurer un résultat d'attention ou d'action unique.

4.1.9 *Instinct* – Réseaux neuronaux pré-établis

L'objectif de ce type de gènes est d'établir des connexions entre neurones au sein du système nerveux des individus. Pour que ces connexions s'établissent à certains stades de la vie de l'individu et pour éviter une concurrence éventuelle avec d'autres constructions neuronales au cours de la vie de l'individu, ce type de gène dépend d'une variable chimique définie qui diminue à l'usage. Les variables de ce type de gènes sont : la variable utilisée, la quantité utilisée, les coordonnées du neurone de départ, les coordonnées du neurone d'arrivée et le poids de la connexion. Si la connexion existe déjà et que la variable chimique est encore présente, la connexion est simplement renforcée.

L'intérêt principal de la présence de ce type de gène est de construire certains comportements quelque soit la complexité du système nerveux de l'organisme : le nombre de gènes de ce type est bien sûr dépendant de la complexité des comportements à obtenir et des concepts au sein du réseau de neurones et des connexions à construire entre neurones.



L'utilisation d'une variable chimique déterminée pour le fonctionnement de ce gène entraîne une dépendance à la présence du composé, de préférence par l'utilisation d'un gène *InitialConcentration* pour un développement des connexion dans les premiers stades de la vie de l'individu ; éventuellement, cela peut intervenir au cours de la vie de l'individu par l'apparition du composé suite un gène *BiochemicalReaction* ou un gène *EmitterReceptor*).

4.2 Liste normalisée des «éléments chimiques»

Par convention, le premier élément de cette liste normalisée (tableau 13) à la position 000 est *<NONE00>*, tous les autres éléments «non attribués» sont nommés de la même façon entre chaque groupe d'éléments définis. Les éléments 001 à 118 symbolisent directement la chimie de base, sans prendre en compte les isotopes éventuels. Les éléments à partir de 151 symbolisent des molécules de biochimie classique (acides aminés, nucléotides, acides gras, sucres...); puis à partir de 600, les hormones, et à partir de 651, les phéromones. Les éléments à partir de 800 sont utilisés par le réseau neuronal et seront détaillé dans le tableau 14.

Nomenclature des variables de chimie et biochimie							
Liste des atomes (table périodique des éléments)							
001	H Hydrogen	031	Ga Gallium	061	Pm Promethium	091	Pa Proctatinium
002	He Helium	032	Ge Germanium	062	Sm Samarium	092	U Uranium
003	Li Lithium	033	As Arsenic	063	Eu Europium	093	Np Neptunium
004	Be Beryllium	034	Se Selenium	064	Gd Gadolinium	094	Pu Plutonium
005	B Bore	035	Br Brome	065	Tb Terbium	095	Am Americium
006	C Carbon	036	Kr Krypton	066	Dy Dysprosium	096	Cm Curium
007	N Azote	037	Rb Rubidium	067	Ho Holmium	097	Bk Berkelium
008	O Oxygen	038	Sr Strontium	068	Er Erbium	098	Cf Californium
009	F Fluor	039	Y Yttrium	069	Tm Thulium	099	Es Einsteinium
010	Ne Neon	040	Zr Zirconium	070	Yb Ytterbium	100	Fm Fermium
011	Na Sodium	041	Nb Niobium	071	Lu Lutécium	101	Md Mendelevium
012	Mg Magnesium	042	Mo Molybdene	072	Hf Hafnium	102	No Nobelium
013	Al Aluminium	043	Tc Technetium	073	Ta Tantale	103	Lr Lawrencium
014	Si Silicium	044	Ru Ruthenium	074	W Tungstène	104	Rf Rutherfordium
015	P Phosphor	045	Rh Rhodium	075	Re Rhenium	105	Db Dubnium
016	S Soufre	046	Pd Palladium	076	Os Osmium	106	Sg Seaborgium
017	Cl Chlore	047	Ag Argent	077	Ir Iridium	107	Bh Bohrium
018	Ar Argon	048	Cd Cadmium	078	Pt Platium	108	Hs Hassium
019	K Potassium	049	In Indium	079	Au Auri – Or	109	Mt Meitnerium
020	Ca Calcium	050	Sn Sin (Etain)	080	Hg Mercure	110	Ds Darmstadtium
021	Sc Scandium	051	Sb Stibium	081	Tl Thallium	111	Rg Roentgenium
022	Ti Titanium	052	Te Tellurium	082	Pb Plomb	112	Cn Copernicium
023	V Vanadium	053	I Iodine (Iode)	083	Bi Bismuth	113	Uut Uut
024	Cr Chrome	054	Xe Xenon	084	Po Polonium	114	Uuq Uuq
025	Mn Manganese	055	Cs Caesium	085	At Astate	115	Upup Upup
026	Fe Fer	056	Ba Barium	086	Rn Radon	116	Uuh Uuh
027	Co Cobalt	057	La Lanthane	087	Fr Francium	117	Uus Uus
028	Ni Nickel	058	Ce Cerium	088	Ra Radium	118	Uuo Uuo
029	Cu Cuivre	059	Pr Praseodyne	089	Ac Actinium	119	<NONE01>
030	Zn Zinc	060	Nd Neodyme	090	Th Thorium	120	<NONE02>
Hormones (600 et plus) et Phéromones (650 et plus)							
370	Hormone00	375	Hormone05	350	Pheromone00	355	Pheromone05
371	Progesterone	376	Hormone06	351	Pheromone01	356	Pheromone06
372	Oestrogene	377	Hormone07	352	Pheromone02	357	Pheromone07
373	Testosterone	378	Hormone08	353	Pheromone03	358	Pheromone08
374	Hormone04	379	Hormone09	354	Pheromone04	359	Pheromone09
Molécules biochimiques complexes (à partir de 151)							
150	<NONE31>	183	H ₂ O	216	AAE Glutamate	249	Acide Pelargonique
151	ATP	184	Hydroxyd (OH ⁻)	217	AAQ Glutamine	250	Acide Caprique
152	ADP	185	AcetylCoA	218	AAG Glycine	251	Acide Undecylique
153	AMP	186	OxaloAcetate	219	AAH Histidine	252	Acide Laurique
154	CTP	187	Citrate	220	AAI IsoLeucine	253	Acide Tridecylique
155	CDP	188	CoASH	221	AAL Leucine	254	Acide Myristique
156	CMP	189	cis-Aconitate	222	AAK Lysine	255	Acide Pentadecyl.
157	GTP	190	iso-Citrate	223	AAM Methionine	256	Acide Palmitique
158	GDP	191	Oxalosuccinate	224	AAF PhenylAlanine	257	Acide Margarique
159	GMP	192	alpha-Cetoglutarate	225	AAP Proline	258	Acide Stearique
160	TTP	193	SuccinylCoA	226	AAO Pyrolysine	259	Acide Nonadecyl.
161	TDP	194	Succinate	227	AAU SelenoCyst.	260	Acide Arachidique
162	TMP	195	Fumarate	228	AAS Serine	261	Acide hénéicosan.
163	UTP	196	Malate	229	AAT Threonine	262	Acide Béhénique
164	UDP	197	CyclicElement02	230	AAW Tryptophane	263	Acide tricosanoique
... Suite page suivante ...							

Nomenclature des variables de chimie et biochimie(suite)							
165	UMP	198	CyclicElement01	231	AAV Tyrosine	264	Acide Lignocerique
166	NAD	199	Pyruvate	232	AAV Valine	265	Acide pentacosan.
167	NADH,H ⁺	200	Adenine	233	<i>AminoAcid00</i>	266	Acide Cérotique
168	FAD	201	Cytosine	234	<i>AminoAcid01</i>	267	Acide heptacosan.
169	FADH2	202	Guanine	235	<i>AminoAcid02</i>	268	Acide Montanique
170	Glucose	203	Thymin	236	<i>AminoAcid03</i>	269	Acide nonacosan.
171	Fructose	204	Uracil	237	<i>AminoAcid04</i>	270	Acide Melissique
172	Glucose-6-P	205	<i>BasicMole00</i>	238	<i>AminoAcid05</i>	271	Acide hentriaco.
173	Fructose-6-P	206	<i>BasicMole01</i>	239	<i>AminoAcid06</i>	272	Acide Laceroique
174	Fructose-1,6-biP.	207	<i>BasicMole02</i>	240	<i>AminoAcid07</i>	273	<i>FattyAcid00</i> (33c)
175	DHAP	208	<i>BasicMole03</i>	241	Acide Formique	274	<i>FattyAcid01</i> (34c)
176	PGAL	209	<i>BasicMole04</i>	242	Acide Acétique	275	<i>FattyAcid02</i> (35c)
177	BisphosphoGlycerate	210	<i>BasicMole05</i>	243	Acide Propionique	276	<i>FattyAcid03</i> (36c)
178	3-PhosphoGlycerate	211	AAA Alanine	244	Acide Butyrique	277	<i>FattyAcid04</i> (37c)
179	2-PhosphoGlycerate	212	AAR Arginine	245	Acide Valerique	278	<i>FattyAcid05</i> (38c)
180	phosphoenolPyruvate	213	AAN Asparagine	246	Acide Caproique	279	<i>FattyAcid06</i> (39c)
181	O ₂	214	AAD Aspartate	247	Acide Enanthique	280	<i>FattyAcid07</i> (40c)
182	CO ₂	215	AAC Cysteine	248	Acide Pelargonique	281	<NONE00>

TABLE 13: Nomenclature des variables de chimie et biochimie

Encodage des directions (800 – 829)					
800	(local) (LLL)	810	Up (UUU)	820	Down (DDD)
801	NorthWest (LNW)	811	(UNW)	821	(DNW)
802	North (LNN)	812	(UNN)	822	(DNN)
803	NorthEast (LNE)	813	(UNE)	823	(DNE)
804	East (LEE)	814	(UEE)	824	(DEE)
805	SouthEast (LSE)	815	(USE)	825	(DSE)
806	South (LSS)	816	(USS)	826	(DSS)
807	SouthWest (LSW)	817	(USW)	827	(DSW)
808	West (LWW)	818	(UWW)	828	(DWW)
Encodage des stimuli et décisions (850 – 880)					
850	Stay	860	Rest	870	Mate
851	Push(x)	861	Sleep	871	(free)
852	Pull(x)	862	Eat	872	(free)
853	Stop	863	Death	873	(free)
854	Move to(x)	864	Emit(x)	874	(free)
855	Move away	865	Receive(x)	875	(free)
856	Get(x)	866	Has(x)	876	(free)
857	Drop(x)	867	Is(x)	877	(free)
858	Think(x)	868	MakeGamet	878	(free)
859	Slap(x)	869	LayEgg	879	RecordState
Encodage des objets et des états					
900	Current	920	Gamet	940	Daemon
901	Small elt	921	Egg	941	Bacta
902	Midd elt	922	Embryo	942	Plant
903	Big elt	923	Larva	943	Anima
904	Food	924	Child	944	Virus
905	Drink	925	“Teen”	945	(free)
906	Vehicle	926	Adult	946	(free)
907	Automaton	927	Senior	947	(free)
908	Computer	928	Dead	948	(free)
909	Laptop	929	Not Acc.	949	(free)
910	Phone	930	(free)	950	Gender (sex)
911	Cell Phone	931	(free)	951	Aging
912	Agent	932	(free)	952	AgentType
913	(free)	933	(free)	953	TypeOf
914	(free)	934	(free)	954	Status
915	(free)	935	(free)	955	Movable
916	(free)	936	(free)	956	Eatable
917	(free)	937	(free)	957	Fertile
918	(free)	938	(free)	958	Pregnant
919	(free)	939	(free)	959	(free)

TABLE 14 – Nomenclature pour les Stimuli et Décisions

Cette partie de la nomenclature concerne une utilisation (interne ou externe) par les gènes de type *Stimulus-Decision* et *EmitterReceptor*. Seules les 9 premières directions sont utilisées. L'action 864 entraîne la mort de l'agent courant, les états à partir de 940 le décrivent.



4.3 Encodage génétique

4.3.1 Gènes «paramètres» – *GeneGattaca*

Dans ce cas les gènes sont des fonctions pré-encodées comme dans *Creatures*, avec des paramètres susceptibles de varier ; les variables sont numérotées de 000 à 999 ; les gènes sont fortement typés (regroupés sous le terme *GeneGattaca*). L'écriture des gènes est définie par l'ordre des paramètres et utilisera le code décrit dans le tableau 15 lors de sauvegarde externes. Ces gènes sont définis comme commençant avec “M” et finissant avec “*”. Afin de permettre une lisibilité humaine, l'idée de base est de les encoder ligne par ligne.

Lors d'un chargement dans le modèle, les gènes créés avec le codage paramétrique sont traduits et reconnus en fonction de leur longueur (nombre de caractères nécessaires pour les paramètres), puis sont chargés en mémoire.

Les paramètres des gènes sont codés selon les modes suivants : les **booléens** sont des chiffres soient pairs (*true* – vrai) soient impairs (*false* – faux), les **valeurs numériques** sont repérées et regroupées linéairement selon le nombre de chiffres nécessaires et traduits pour dans le sens de lecture (de gauche à droite) par des puissances décroissantes de dix, jusqu'à l'unité. Dans le cas d'un paramètre définis par un nombre à trois chiffres, le premier chiffre lu est celui des centaines, puis le second celui des dizaines, et enfin celui des unités.

Le code est redondant pour les chiffres car ce les plus utilisés dans les paramètres, les lettres sont là pour construire les annotations avec les séparateurs : et '.

aaa	a	caa	q	taa	2	gaa	7
aac	b	cac	r	tac	2	gac	7
aat	c	cat	s	tat	2	gat	8
aag	d	cag	t	tag	3	gag	8
aca	e	cca	u	tca	3	gca	8
acc	f	ccc	v	tcc	3	gcc	9
act	g	cct	w	tct	4	gct	9
acg	h	c cg	x	tcg	4	gcg	9
ata	i	cta	y	tta	4	gta	:
atc	j	ctc	z	ttc	5	gtc	:
att	k	ctt	0	ttt	5	gtt	'
atg	l	ctg	0	ttg	5	gtg	'
aga	m	cga	0	tga	6	gga	M
agc	n	cgc	1	tgc	6	ggc	M
agt	o	cgt	1	tgt	6	ggt	*
agg	p	cgg	1	tgg	7	ggg	*

M indique les codons start et * les codons stop.

TABLE 15 – Les 64 triplets du codage paramétrique *GeneGattaca*

L'en-tête des gènes (**header**) est défini pour l'ensemble des types de gènes avec les paramètres suivants (15 caractères) :

- mutate (booléen)
- duplicate (booléen)
- delete (booléen)
- activ (booléen)
- ageMin (000 à 999)
- ageMax (000 à 999)
- sex (000 à 9999)
- mutRate (00 à 99)

Les types de gènes créés avec ce code (*GeneGattaca*) sont décrits, avec leurs paramètres et valeurs possibles, dans le tableau 16. Un attribut complémentaire (*more*) a été ajouté au *BrainGene* pour éviter une confusion avec *InitialConcentration* dans la longueur du gène décrit.

D'autres types de gènes peuvent être développés au-delà de cette base de gènes paramétriques, comme les gènes de transposition ou les gènes viraux. L'idée initiale de ces gènes est de provoquer des comportements particuliers au niveau des génomes et de la biochimie des organismes modélisés. Dans le cas des transposons, il suffit de déplacer des gènes au sein du génome d'un organisme : l'ordre d'exécution des gènes dans le modèle peut avoir un avantage évolutif. Concernant les gènes viraux, l'objectif est le détournement du métabolisme hôte pour produire de nouvelles particules virales. **Dans ces deux cas, gènes viraux et transposons, en plus des gènes déjà établis, un seul type de gène paramétrique est à ajouter : des gènes de modification du génome au niveau des gènes (insertion, délétion, réarrangement).**

4.3.2 Gènes «encodage complet»

La traduction donne un code exécutable du type Java ou C/C++ (à tester avant d'exécuter, ceci favorise et provoque une sélection sur l'ensemble du code). Le choix de certains éléments de programmation dans le codage a été fait pour éviter une trop grande variation unitaire en leur sein lors de mutations. Un gène de contrôle «Hello world» est possible (et n'induit pas de critère de sélection à-priori).

Voir le tableau 17 : *Redondance minimale des caractères alphabétiques (majuscule et minuscule), ainsi que des opérateurs et parenthèses. Un peu plus de redondance pour les codons de «code source» et des codons stop éventuels de signal de fin du gène. \n pour les retour-chariot; \t pour les tabulations et \s pour les espaces simples.*



Type	Longueur	Paramètre	Valeurs
InitialConcentration	6 (2*3)	variable	[000-999]
		value	[000-999]
BiochemicalReaction	27 (9*3)	Achem	[000-999]
		Acoef	[000-999]
		Bchem	[000-999]
		Bcoef	[000-999]
		Cchem	[000-999]
		Ccoef	[000-999]
		Dchem	[000-999]
		Dcoef	[000-999]
		KMVM	[000-999]
BrainGene	8 (4*2)	height	[00-99]
		width	[00-99]
		depth	[00-99]
		more	[00-99]
BrainLobeGene	32 (7*3+4*2+3)	rest	[000-999]
		threshold	[000-999]
		descent	[000-999]
		dendr_min	[000-999]
		dendr_max	[000-999]
		proximity	[000-999]
		reproduction	boolean
		repro_num	[000-999]
		WTA	boolean
		height	[00-99]
		width	[00-99]
		posx	[00-99]
		posy	[00-99]
		replace	boolean
EmitterReceptor	15 (3*3+2*2+2)	variable	[000-999]
		threshold	[000-999]
		put	[000-999]
		posx	[00-99]
		posy	[00-99]
		receptor	boolean
		internal	boolean
StimulusDecision	20 (2+6*3)	perception	boolean
		object	boolean
		indicator	[000-999]
		threshold	[000-999]
		attribute	[000-999]
		variable	[000-999]
		value	[000-999]
		script	[000-999]
Instinct	17 (3*3+4*2)	variable	[000-999]
		threshold	[000-999]
		inputPosX	[00-99]
		inputPosY	[00-99]
		outputPosX	[00-99]
		outputPosY	[00-99]
		weight	[000-999]
		check	boolean
		isPositive	boolean

TABLE 16 – Description brève des types de gènes paramétriques

Les valeurs possibles des attributs sont soit des booléens, soit des entiers entre 0 et 99 (lié au réseau de neurones) ou entre 0 et 999 (lié aux variables).

uuuu	a	uvuu	q	buuu	G	bvu	W	vu	6	vvu	?	puu	class	pvu	double
uub	a	uvub	q	bub	G	bvub	W	vub	6	vvub	?	puub	class	pvub	double
uuuv	b	uvuv	r	buv	H	bvu	X	vuv	6	vvuv	:	puuv	class	pvuv	double
uup	b	uvup	r	bup	H	bvup	X	vup	6	vvup	:	puup	class	pvup	double
uubu	c	uvbu	s	bubu	I	bvu	Y	vbu	7	vvbu	++	pubu	interface	pvbu	true
uubb	c	uvbb	s	bubb	I	bvbb	Y	vbb	7	vvbb	++	pubb	interface	pvbb	true
uubv	d	uvbv	t	bubv	J	bvv	Z	vubv	7	vvbv	--	pubv	interface	pvbv	true
uubp	d	uvbp	t	bubp	J	bvbp	Z	vubp	7	vvbp	--	pubp	interface	pvbp	true
uuvu	e	uvvu	u	buvu	K	bvu	0	vuvu	8	vvvu	;	puvu	static	pvvu	false
uuvb	e	uvvb	u	buvb	K	bvvb	0	vuvb	8	vvvb	;	puvb	static	pvvb	false
uuvv	f	uvvv	v	buvv	L	bvvv	0	vuvv	8	vvvv	\n	puvv	static	pvvv	false
uuvp	f	uvvp	v	buvp	L	bvvp	0	vuvp	8	vvvp	\n	puvp	static	pvvp	false
uupu	g	uvpu	w	bupu	M	bvpu	1	vupu	9	vvpu	\t	pupu	final	pvpu	if
uupb	g	uvpb	w	bupb	M	bvpb	1	vupb	9	vvpb	\t	pupb	final	pvpb	if
uupv	h	uvpv	x	bupv	N	bvpv	1	vupv	9	vvpv	\s	pupv	final	pvpv	else
uupp	h	uvpp	x	bupp	N	bvpp	1	vupp	9	vvpp	\s	pupp	final	pvpp	else
ubuu	i	upuu	y	bbuu	O	bpu	2	vbu	=	vpu	(pbuu	private	ppuu	while
ubub	i	upub	y	bbub	O	bpub	2	vub	=	vpub	(pbub	private	ppub	while
ubuv	j	upuv	z	bbuv	P	bpuv	2	vbu	!	vpuv)	pbuv	private	ppuv	while
ubup	j	upup	z	bbup	P	bpu	2	vbu	!	vpu)	pbup	private	ppup	while
ubbu	k	upbu	A	bbbu	Q	bpbu	3	vbbu	.	vpbu	[pbbu	public	ppbu	&&
ubbb	k	upbb	A	bbbb	Q	bpb	3	vbbb	.	vpb	[pb	public	ppb	&&
ubbv	l	upbv	B	bbbv	R	bpbv	3	vbbv	-	vpbv]	pbv	public	ppv	
ubbp	l	upbp	B	bbbp	R	bpbp	3	vbbp	-	vpbp]	pb	public	ppp	
ubvu	m	upvu	C	bbvu	S	bpvu	4	vbbu	+	vpvu	{	pbvu	void	ppvu	return
ubvb	m	upvb	C	bbvb	S	bpbv	4	vbbv	+	vpbv	{	pbv	void	ppv	return
ubvv	n	upvv	D	bbvv	T	bpvv	4	vbbv	/	vpvv	}	pbvv	void	ppvv	return
ubvp	n	upvp	D	bbvp	T	bvpv	4	vbbp	/	vpvp	}	pbvp	void	ppvp	return
ubpu	o	uppu	E	bbpu	U	bppu	5	vbbu	*	vppu	<	pbpu	int	pppu	[STOP]
ubpb	o	uppb	E	bbpb	U	bppb	5	vbbp	*	vppb	<	pbpb	int	pppb	[STOP]
ubpv	p	uppv	F	bbpv	V	bppv	5	vbbv	"	vppv	>	pbpv	int	pppv	[STOP]
ubpp	p	uppp	F	bbpp	V	bppp	5	vbbp	"	vppp	>	pbpp	int	pppp	[STOP]

TABLE 17 – Tableau des 256 quadruplets standards du codage «complet»



4.4 Modélisation et conception d'agents – Construction de tests

4.4.1 Entretien de l'environnement : les *Daemons*

De façon à maintenir des ressources en quantité suffisantes pour l'ensemble des organismes présent dans l'environnement, une catégorie particulière d'agents intervient pour gérer les ressources (renouvellement, répartition) et évacuer les organismes «morts». Il s'agit des *Daemons* et sont équivalent à l'idée d'un ramasse-miette (ou *garbage collector*) de la machine virtuelle JAVA ou d'un système d'exploitation.

Ces agents sont codés génétiquement afin d'interagir de façon similaire aux autres agents «vivants» dans l'environnement, certaines actions leur étant réservées. Un genre spécifique leur est réservé pour indiquer leur type : *Silicodaemon*.

4.4.2 Organismes de bases : bactéries et plantes

Les genres *Silicobacter* et *Silicoviriditae* représentent des types particuliers d'organismes : les bactéries et les plantes. Ces types d'agents ne se déplacent pas à priori dans l'environnement mais peuvent se reproduire (scissiparité et production de fruits). L'intérêt de modéliser ces deux types est de permettre d'implémenter des agents représentant des organismes simples, limités à une biochimie et sans système nerveux.

Les bactéries sont notamment utiles pour créer une biochimie et composer des chaînes de réactions métaboliques et tester les comportements possibles. Les plantes sont utiles pour les comportements de production d'objets ou d'autres agents (des fruits, pouvant devenir des plantes) utilisables par d'autres agents différents comme sources de nourriture.

4.4.3 Agent en déplacement

En plus de ce que font les *daemons* dans l'environnement, l'objectif est ici de classer les espèces selon leur mode de déplacement à la manière des pièces d'un jeu d'échecs (dans la limite d'un déplacement d'une case à la fois). L'intérêt est d'observer un comportement d'ensemble en fonction des ressources et de l'évolution des espèces.

4.4.4 Espèces sexuées

Une expérience ensuite est de mettre en place des espèces du genre *Silicoanimae* pour disposer de modèles d'espèces disposant d'une reproduction sexuée, produisant des gamètes et utilisant ces derniers pour produire des œufs ou de nouveaux individus. La fusion de gamètes peut être étudiée plus précisément (définition d'une espèce par le nombre de chromosomes, de gènes...); afin notamment de savoir quels sont, parmi les gamètes produits, ceux qui produisent des individus viables et / ou proches de leurs ascendants. Et également pour voir l'impact d'une reproduction parthénogénétique.

4.4.5 Virus, système immunitaire : impact...

Une étude peut être menée dans l'introduction de génomes viraux dans les organismes ou, plus simplement, l'introduction de particules virales dans l'environnement. L'impact sur la survie des organismes est intéressante à plusieurs titres (à comparer avec des modèles réels; réactions des organismes; mise en place d'un système immunitaire; ...)

4.4.6 Écosystème complexe

Plusieurs écosystèmes peuvent être mis en place et étudiés, suite aux résultats des tests des points précédents (bactéries, plantes, espèces animales, virus...); on peut notamment prendre en compte un système dit «complet», comprenant au moins un cycle proie / prédateur ou la présence conjointe de plusieurs espèces concurrentes pour une même source de nourriture / d'énergie. Un tel modèle est à étudier sur différents impacts, comprenant la proportion relative des individus des différentes espèces, leur sélection et leur évolution séparée ou conjointe sur plusieurs générations.

D'autres modèles sont à étudier : fourmilière et assimilés (insectes sociaux : répartition des rôles, installation du nid, récupération de la nourriture...). Leur étude peut servir à trouver des algorithmes d'organisations optimaux, tant au niveau social (coopération entre agents / organismes) que génétique (génétique de la population de la fourmilière).

4.4.7 Modèle complexe : «Test de la fourmilière»

On constate par l'observation de la nature (en particulier des insectes) que la juxtaposition de comportements simples engendre parfois un comportement global «intelligent». Les systèmes multi-agents s'inspirent de cette observation : des agents autonomes très simples collaborent de sorte à résoudre une tâche complexe qu'aucun d'eux ne pourrait réaliser seul. La grande force du système est que la tâche globale n'est jamais décrite de façon explicite, mais apparaît au contraire spontanément de l'agrégation des comportements individuels.

On se propose d'étudier un système multi-agents simple : une fourmilière. Chaque fourmi est capable de chercher de la nourriture de façon basique : quand elle voit dans son voisinage immédiat de la nourriture, la fourmi prend la nourriture et retourne à la fourmilière (dont elle connaît toujours la direction) pour déposer la nourriture transportée ; elle retourne ensuite au travail. Si elle ne voit pas de nourriture dans son voisinage immédiat, elle se déplace de façon aléatoire, afin d'explorer son environnement.

Le but global est que la fourmilière soit approvisionnée le plus rapidement et le plus efficacement possible, ce qui n'est pas le cas en général quand le déplacement des fourmis est complètement aléatoire. Dans la nature, le problème est résolu par l'utilisation de phéromones. Quand une fourmi a trouvé de la nourriture, elle dépose des phéromones sur son trajet de retour à la fourmilière. Dans leur déplacement aléatoire, les fourmis sont attirées par les phéromones et ont donc tendance à suivre des chemins mis en place par les pionnières pour atteindre les «réserves» de nourriture. Le programme à réaliser doit simuler ce comportement.

Modélisation

- **de l'environnement** : un «monde» avec des cases pouvant contenir des objets et contenant un ensemble de variables chimiques (phéromones par exemple).
- **des agents (plantes, fruits, fourmis)** : une définition générique d'«Agent» pouvant être exécuté dans sa programmation et contenant des variables chimiques. La programmation reprend les concepts génétiques précédemment décrits dans les idées (gènes : concentrations initiales, réactions biochimiques, émetteurs, récepteurs, stimuli, instincts...).

L'objectif de ce modèle de fourmilière est de fournir un outil pour travailler à l'élaboration des éléments nécessaires pour un modèle général, d'en assurer le développement ainsi que d'en donner des exemples d'implémentation de plusieurs gènes dans un même «agent vivant», leurs interactions et leurs effets.



4.5 Interfaces Homme-Machine

4.5.1 Conception de gènes et voies métaboliques

GeneCreator est le nom de cette interface représentée dans le tableau 18. L'intérêt de cet interface graphique est de créer et de manipuler des gènes et des voies métaboliques, afin de pouvoir les intégrer à une voie métabolique où, par la suite, à un organisme.

Select defined Gene <List of Gene's>	<Choose Type of Gene>	Create Gene	Select defined gene <List of Gene's>
	Change Gene	As new Gene	
	<Gene Form>		
	Gene Name :		
	List of Chemicals :	<List of chemicals>	
<i>Delete Gene</i>			<i>Remove Gene</i>
Existent Pathways <Select Metabolic Path>			Path Name : < > <i>Add gene to Pathways</i> <i>Create PathWay</i>

TABLE 18 – L'interface *GeneCreator*

Description des différents éléments et leur rôle.

- *Choose Type of Gene* : sélection / affichage du type de gène.
- *Create Gene* : créer un gène du type sélectionné.
- *Change Gene* : modifier le gène sélectionné.
- *As new Gene* : enregistrer le gène sélectionné comme nouveau.
- *Gene Name* : indiquer le nom du gène.
- *List of Chemicals* : liste des éléments chimiques.
- *Select defined Gene* : liste de gènes déjà définis (sélection et affichage).
- *Existent Pathways* : liste de voies métaboliques déjà définies.
- *Colonne de droite* : définition de voies métaboliques (liste de gènes, nom, ajout et suppression de gène...).

Description de l'affichage des différents Formulaire de type de gène.

- Paramètres généraux pour tous les gènes, affichage et modification de l'entête (*Gene header*) :
 - Mutation, Duplication, Deletion, Actif : cases à cocher (oui par défaut).
 - Age minimal (0), Age maximal (0 / 999), sexe (0 : tous).
 - Taux de mutation (25% par défaut).
- Paramètres spécifiques pour chaque gène, avec leurs valeurs par défaut entre parenthèses :
 - *InitialConcentration* : Chemical index (000), Chemical value (000).
 - *BiochemicalReaction* : [A-D]chem (000), [A-D]coef (000), KM (000).
 - *BrainGene* : Height (00), Width(00), *Depth* (01), *More* (01).
 - *BrainLobeGene* : replaceLobe (non), WTA (non), reproduceNeurone (non, 0), rest (000), threshold (000), descent (000), proximity (000), Height (00), Width (00), X-pos (00), Y-pos (00), Minimal dendrites (00), Maximal dendrites (00).
 - *EmitterReceptor* : Chemical index (000), Threshold (000), X-pos (00), Y-pos (00), IN/OUput (000), Receptor (oui), Internal (oui).
 - *StimulusDecision* : Indicator (000), Threshold (000), Perceptor (oui), Object (oui), Attribute (000), Chemical (000), Value (000), Script (000).
 - *Instinct* : Origin X-pos (00), Origin Y-pos (00), Destination X-pos (00), Destination Y-pos (00), Weight (000), check (oui), Chemical (000), Threshold (000).

4.5.2 Conception d'organismes (généétique)

GeneticKit est le nom de cette interface présentée dans le tableau 19. L'objectif de cette interface graphique est de manipuler (créer, modifier) des Agents Organismes selon une vue orientée génétique : ajout et suppression de gènes, ajout de voies métaboliques (suite de gènes).

	Open Agent File...	<organism Type> <i>Create an Agent</i>	Name :
Genes List <List of Gene's>	<Select a defined Gene>		
	<Choose type Of Gene>	Create Gene	Add gene to Agent
	<Gene Form>		
	<Select metabolic way>		Add metabolic way to Agent
Remove Gene			
	Save		

TABLE 19 – L'interface *GeneticKit*

Description des différents éléments et leur rôle.

- *Reprise d'éléments de manipulation du GeneCreator, comportement identique.*
- *Ouverture, création sélection du type, nommage.*

4.5.3 Conception avancée d'organismes (phylogénie et documentation)

OrganismKit est le nom de cette interface présentée dans le tableau 20. Cette interface graphique permet une manipulation plus avancée du génome (positionnement des gènes, segmentation en chromosomes), et permet également de gérer le lignée de l'espèce / de l'organisme, ainsi que l'attribution et la modification de noms.

	Open Agent File...	<organism Type> <i>Create an Agent</i>	Name :
Genes List <List of Gene's>			
	Scientific Name	<>	<Lineage List>
	BioSilico Name	<>	
	Common Name	<>	
	Include	<>	
	Others Names	<other Names List>	
<>	<other Names List>		
<i>Add other name</i>	<other Names List>		
<i>Up Gene – Down Gene</i> <i>Add Chromosome</i>			<i>Add Lineage – Remove Lineage</i> <i>Up Lineage – Down Lineage</i>
	Save		

TABLE 20 – L'interface *OrganismKit*

Description des différents éléments et leur rôle.

- *Espace de nommage (noms scientifiques, d'usage...).*
- *Ouverture, création sélection du type, nommage simple.*
- *Déplacement des gènes, ajout indication de séparateur de chromosome.*
- *Modification de lignée (espèce, organisme).*



4.5.4 Autres outils (interaction)

Jusqu'ici les outils implémentés avaient pour objectif de créer et définir les organismes et leurs composantes génétiques. L'objectif ensuite est de fournir des outils d'interaction avec l'environnement modélisé, dont les fonctionnalités seraient les suivantes :

- fournir des rapports d'états environnementaux (général ou discret),
- fournir des données sur un organisme modélisé (biochimie, réseau neuronal, génétique...),
- introduire / exporter un organisme.

Différents outils (*Kits*) peuvent être construits sur un modèle générique :

1. Kit d'Observation (vue générale des organismes et de l'environnement)
 - Liste des organismes avec leur localisation, et quelques informations d'états (age, sexe...).
 - Zones de discretisation de l'environnement et leur «climat», nombre d'individus...
 - Alertes configurables par l'utilisateur.
2. Kit Scientifique (étude des organismes et interactions)
 - Sélection d'un organisme.
 - Aspects génétiques, biochimiques, phylogénie, ascendance / descendance...
 - Visualisation du fonctionnement du réseau neuronal (anatomie, processus décisionnel...).
 - Injection / retraits d'éléments chimiques.
3. Kit Communication
 -
4. Kit Croisement
 - Croiser des organismes de même Famille / Genre / Espèces.
 - A partir d'archives ou d'individus existants.
 - Choix du stade de vie de l'organisme.
 - Génétique «croisement naturel» ou choix.
 - Arbres généalogiques, calcul du coefficient de consanguinité.

5 Reprise du projet en 2020

5.1 Sortie de sommeil du projet

Ce projet a été mis en veille pendant un certain temps (une petite dizaine d'année ; diverses circonstances ont amené à le reprendre, autant par passion inoubliable que l'intérêt d'application autres que la simple simulation et modélisation (utilisation en analyse de données, jeu vidéo, utilisation d'agents autonomes «intelligents» au sein d'un système informatique...).

Un résumé des quelques éléments repris, modifiés, ajoutés (juillet-septembre 2020) :

- reprise du code pour le rendre plus professionnel et industriel :
 - tests unitaires systématiques sur le code existant,
 - relecture du code (*review*),
 - généralisation de certaines parties (Interfaces IChemicals, IEnvironment, IEnvironmentItem...) : l'idée étant de pouvoir réutiliser certains éléments ailleurs,
 - partitionnement via *Maven* entre parties indépendantes rendre ce projet plus lisible,
 - intégration de parties jusqu'ici séparées et / ou faites indépendamment mais en lien avec ce projet (chiffrement, modélisation cellulaire, documentation et reverse-ingénierie dur *Creatures*, tests sur des réseaux de neurones, volonté de tester *BioJava*...)
 - ...
- revue et correction de la liste des «éléments chimiques» (Chemicals) utilisés (ajouts et changements utiles) ;
- des idées complémentaires concernant les *Daemons* (agents utilitaires), les gènes viraux (transposition mais aussi production de particules virales) ;
- modélisation à partir d'un système de voies métaboliques minimales (repris de *Creatures*) afin d'effectuer des premiers tests plus facilement (une fourmi et une plante, une source d'énergie, un fournisseur d'éléments chimiques basiques et un convertisseur de fruits) ;
- une modélisation des cycles existant connus (Krebs, Lynen...) est possible et à étudier / préciser ;
- idées d'utilisation de ce projet : agents autonomes dans un modèle informatique et / ou un environnement logiciel (analyse de logs, analyses de machines, analyses de réseau informatique...), et également dans un univers de type MMORPG / Jeu Vidéo (génération procédurale, animaux et plantes avec des comportements cohérents...) ;
- ...


5.2 Publication GitHub et autres idées

Ce projet est sur la plate-forme GitHub depuis la fin juin 2020, pour différentes raisons :

- Suivi de version du code (évolution de certaines parties...) ;
- Rendre ce projet public ;
- Obtenir des retours éventuels ;
- Faire participer d'autres personnes ;
- Pouvoir intégrer d'autres idées et outils ;
- Intégrer des cas d'utilisation de BioJava (et éventuellement BioPerl et / ou BioPython ?)
- ...



Abréviations

BLAST	Basic Local Alignment Search Tool <i>algorithmes et logiciels pour l'alignement de séquences et la recherche de similarités locales</i>
BLOSUM	BLOCK SUBstitution Matrix <i>Matrice de substitution de calcul de score lors d'alignement de séquences (domaines)</i>
CDR	The Creatures Developer Resource
	GLIDER est le logo des hacker, symbole repris du jeu de la vie (cavalier ou planeur).
NJ	Neighbour-Joining <i>Méthode d'analyse de texte et de regroupement par similarités en clade et arbres</i>
NW	Needleman et Wunsch <i>Algorithme d'alignement global de séquences</i>
PAM	Point Accepted Mutation <i>Matrice de substitution de calcul de score lors d'alignement de séquences (familles)</i>
RQ	Red Queen <i>Course à l'évolution ; système de co-évolution.</i>
SW	Smith et Waterman <i>Algorithme d'alignement local de séquences</i>
UPGMA	Unweighted Pair Group Method with Arithmetic mean <i>Méthode d'analyse de texte et de regroupement par similarités en clade et arbres</i>
WTA	Winner Take All <i>Comportement d'ensemble d'un groupe de neurones où le plus actif d'entre eux reste actif et les autres sont mis à zéro.</i>



Bibliographie

Références

- [1] James Mark BALDWIN : A new factor in evolution. *The American Naturalist*, 30(354):441–451, June 1896.
- [2] Nick BARTON et Willem ZUIDEMA : Evolution : the erratic path towards complexity. *Current Biology*, 13(16):R649–R651, Aug 2003.
- [3] Greg BEAR : *L'échelle de Darwin*. Collection Ailleurs&Demain (Robert Laffont), 2001.
- [4] Greg BEAR : *Les enfants de Darwin*. Collection Ailleurs&Demain (Robert Laffont), 2003.
- [5] B. BENES et E.D ESPINOSA : *Modeling virtual ecosystems with the proactive guidance of agents*, pages 126–131. 16th International Conference on Computer Animation and Social Agents, 2003., Campus Ciudad de Mexico, ITESM, Mexico, May 2003.
- [6] Bedrich BENES et Juan Miguel Soto GUERRERO : Clustering in virtual plant ecosystems. In Copyright UNION Agency Science PRESS, éditeur : *WSCG'2004, February 2-6, 2004, Plzen, Czech Republic*. WSCG SHORT Communication papers proceedings, 2004.
- [7] CHRIS@DOUBLE.CO.NZ : The creatures developer resource – <http://www.double.co.nz/creatures/>.
- [8] Dave CLIFF et Stephen GRAND : The Creatures Global Digital Ecosystem. *Artificial Life*, 5:77–94, 1999. 1999 Massachusetts Institute of Technology.
- [9] Dave CLIFF et Geoffrey F. MILLER : Tracking The Red Queen : Measurements of adaptive progress in co-evolutionary simulations. In *Lecture Notes In Computer Science – Advances in Artificial Life*, volume 929, pages 200–218. Proceedings of the Third European Conference on Advances in Artificial Lif, Springer Verlag, 1995.
- [10] Kerstin DAUTENHAHN : The Art of Designing Socially Intelligent Agents - Science, Fiction and the Human in the Loop. *Applied Artificial Intelligence Journal, Special Issue on Socially Intelligent Agents*, 12:12–17, 1998.
- [11] Marc DUFRÊNE et Pierre LEGENDRE : Species assemblages and indicator species :the need for a flexible asymmetrical approach. *Ecological Monographs*, 67(3):345–366, August 1997.
- [12] ELEMKAY : genornics – <http://meliweb.net/creatures/>.
- [13] Ernesto GERMÁN-SOTO, Leonid B. SHEREMETOV et Christian SÁNCHEZ-SÁNCHEZ : Interaction modeling with artificial life agents. In ITESM Campus Ciudad de Mexico Mexico City D.F Mexico April 27th 2004, éditeur : *Intelligent Virtual Environments and Virtual Agents (IVEVA)*, volume 97, April 2004.
- [14] Stephen GRAND et Dave CLIFF : Creatures : Entertainment software agents with artificial life. *Autonomous Agents and Multi-Agent Systems*, 100:1–20, 1997.
- [15] Stephen GRAND et Dave CLIFF : Creatures : Entertainment software agents with artificial life. *Autonomous Agents and Multi-Agent Systems*, 1:39–57, 1998.
- [16] Stephen GRAND, Dave CLIFF et Anil MALHOTRA : Creatures : Artificial life autonomous software agents for home entertainment. Presentation 9601 / CSRP434, Millennium Technical Report 9601 ; University of Sussex Technical Report CSRP434, 1996.
- [17] Stephen GRAND, Dave CLIFF et Anil MALHOTRA : Creatures : Artificial life autonomous software agents for home entertainment. In *Agents '97 Conference Proceedings*, Proceedings of the first international conference on Autonomous agents, pages 22–29, Marina del Rey, California, United States, 1997. Millenium Interactive, International Conference on Autonomous Agents.
- [18] Inman HARVEY : Evolution : Creatures from another world. *Nature*, 400:618–619, Aug 1999.
- [19] J. E. HOKKANEN : Visual simulations, artificial animals and virtual ecosystems. *The Journal of experimental biology*, 202(Pt 23):3477–3484, Dec 1999.
- [20] Limor ISSACHAROFF, Lubos BUZNA et Dirk HELBING : Overview on bio-inspired operation strategies (deliverable d 2.2.3). Integrated Project (IST Project 027568) 027568, Integrated Risk Reduction of Information-based Infrastructure Systems (IRIIS), September 2007.

- [21] Hiba KHELIL, Abdelkader BENYETTOU et Abdel BELAÏD : Application du système immunitaire artificiel pour la reconnaissance des chiffres. *In Maghrebian Conference on Software Engineering and Artificial Intelligence - MCSEAI'08 (2008)*, December 2008.
- [22] Lemont B. KIER, Paul G. SEYBOLD et Chao-Kun CHENG : *Modeling Chemical Systems Using Cellular Automata – A textbook and laboratory manual*. Numéro 978-1-4020-3657-6 (Print) 978-1-4020-3690-3 (Online). Springer Netherlands, 2005.
- [23] Michael P. KUBE-McDOWELL : *Projet Diaspora (The Quiet Pool)*, volume 3496. J'ai Lu SF, 2003 ?
- [24] Pierre LEGENDRE : Spatial autocorrelation : Trouble or new paradigm ? *Ecology*, 74:1659–1673, September 1993.
- [25] Pierre LEGENDRE et Louis LEGENDRE : *Numerical Ecology*, volume 12–13. Elsevier, 2nd édition, 1998 – 2003.
- [26] Richard E. LENSKI, Charles OFRIA, Robert T. PENNOCK et Christoph ADAMI : The evolutionary origin of complex features. *Nature*, 423(6936):139–144, May 2003.
- [27] Emmanuelle LERAT, Carène RIZZON et Christian BIÉMONT : Sequence divergence within transposable element families in the drosophila melanogaster genome. *Genome Research*, 13(8):1889–1896, Aug 2003.
- [28] Leslie M. LOEW et James C. SCHAFF : The virtual cell : a software environment for computational cell biology. *Trends in biotechnology*, 19(10):401–406, Oct 2001.
- [29] Laurent MIGNONNEAU et Christa SOMMERER : Creating artificial life for interactive art and entertainment. *Leonardo*, 34(4):303–307, August 2001.
- [30] Melanie MITCHELL et Stephanie FORREST : Genetic algorithms and artificial life. *Artificial Life*, 1:267–289, 1993.
- [31] Jean-Philippe RENNARD : <http://www.rennard.org/>.
- [32] Carène RIZZON, Christian BIÉMONT et Manolo GOUY : GATE : un modèle orienté objet dédié à l'analyse des éléments transposables des génomes eucaryotes // gate : an object-oriented model dedicated to the analysis of transposable elements in eukaryote genomes. *In Informatique Mathématiques JOURNÉES OUVERTES BIOLOGIE*, éditeur : *Journées Ouvertes Biologie, Informatique Mathématiques*, pages 301–307, Saint-Malo, Juin 2002. Journées Ouvertes Biologie, Informatique Mathématiques.
- [33] Carène RIZZON, Gabriel MARAIS, Manolo GOUY et Christian BIÉMONT : Recombination rate and the distribution of transposable elements in the drosophila melanogaster genome. *Genome Research*, 12(3):400–407, Mar 2002.
- [34] Boris M. SLEPCHENKO, James C. SCHAFF, John H. CARSON et Leslie M. LOEW : Computational cell biology : spatiotemporal simulation of cellular events. *Annual review of biophysics and biomolecular structure*, 31:423–441, 2002.
- [35] Boris M. SLEPCHENKO, James C. SCHAFF, Ian MACARA et Leslie M. LOEW : Quantitative cell biology with the virtual cell. *Trends in cell biology*, 13(11):570–576, Nov 2003.
- [36] Timothy John TAYLOR : *From Artificial Evolution to Artificial Life*. Thèse de doctorat, University of Edinburg, UK, 1999.
- [37] René THOMAS et Richard D'ARI : *Biological Feedback*. CRC Press, Inc., 1990. 316 page book published in 1990. Mis en ligne avec l'aimable autorisation de l'éditeur.
- [38] WIKIPEDIA : http://en.wikipedia.org/wiki/Aurelia_and_Blue_Moon.
- [39] WIKIPEDIA : http://en.wikipedia.org/wiki/Cellular_automata.
- [40] WIKIPEDIA : http://en.wikipedia.org/wiki/Genetic_algorithm.
- [41] WIKIPEDIA : <http://fr.wikipedia.org/wiki/Parthenogenese>.
- [42] Stephen WOLFRAM : A new kind of science. *Applied Mechanics Reviews*, 56(2):B18–B19, March 2003. ISBN I-57955-088-8 for the book.

