CoCiX

Essais de logiciel simulant une vie artificielle

Cyril LORIN

20 février 2018

Table des matières

Chapitre 1

Introduction & Généralités

1.1 Concept

Nous allons créer un *Objet-CoCiX simulant* un organisme vivant :

Le **CoCiX**

Cet *Objet-CoCiX* aura des *paramètres* représentés dans le programme par des *attributs* de cet *Objet.* (génome, santé, identité, age etc. . .) et des *actions*, représentés par des *Méthodes* (Manger, Dormir, se Déplacer, se Reproduire etc. . .)

Elle aura des besoins vitaux et devra les satisfaire. (Manger, Dormir, se reproduire etc...).

Le logiciel devra faire "vivre" chaque Objet-CoCiX d'une façon autonome, tout en reproduisant les interactions entre les différents individus.

L'utilisateur pourra étudier et voir vivre ses **CoCiX** , il pourra voir l'évolution des générations de **CoCiX** et de leur patrimoine génétique. Il aura la possibilité de connaître à n'importe quelle moment, leur santé et leurs fonctions vitales et ce qu'elles font. Il pourra en outre, rajouter ou enlever des **CoCiX** et modifier leurs environnements.

1.2 Logiciel et langages

1.2.1 Les modules

Le logiciel sera composé de plusieurs modules indépendants :
— Un module de l'utilisateur.
— Un module contrôlant les **CoCiX** .
— Un module gérant l'environnement.
— Un module de statistiques.

1.2.1.1 Module Utilisateur

Ce module doit permettre à l'utilisateur de gérer sa 'colonie' de **CoCiX** . Il doit permettre de voir l'évolution des générations sur la carte Monde, mais aussi pouvoir modifier son environnement, ainsi que les états des **CoCiX** .

C'est dans ce module que l'utilisateur pourra modifier la vitesse du serveur, permettant ainsi de faire évoluer sa colonie plus ou moins vite.

1.2.1.2 Module CoCiX

Module Central du projet, ce module est autonome. Il gère l'ensemble de la base de données des **CoCiX** . Le module va prendre, une part une, les **CoCiX** vivants, et leur faire vivre une durée déterminée par l'utilisateur.

1.2.1.3 Module Environnement

Ce module s'occupe de la base de donnée **Monde** (voir 7.3 page 46). Il doit mettre à jour les informations météorologiques. L'utilisateur pourra l'utiliser pour changer certaines caractéristiques comme *la température*, *l'hydrométrie*, *la radiation* etc . . .d'une région de la carte.

1.2.1.4 Module Statistiques

Ce module doit permettre à l'utilisateur, d'étudier les évolutions statistiques de sa colonie, tant du point de vue *systémique* (Évolution des états généraux des **CoCiX**) que *génétique* (Évolution du Génome et des maladies liées)

1.2.2 Les langages

Les programmes et modules faisant vivre les **CoCiX** ainsi que les différents cycles, seront écrit en **Programmation Objet**. La Base de Données sera gérée en interne.

Les programmes seront hébergés sur un serveur personnel, ce serveur fonctionnant en permanence 24h/24, les utilisateurs devront se connecter sur un compte pour accéder à la gestion de leurs **CoCiX** . La gestion et la consultation se fera par l'intermédiaire de page en html/php sur un serveur **APACHE**.

Première partie

CoCiX

Chapitre 2

Les CoCiX

2.1 Les *Paramètres* d'un CoCiX

Dans cette section, nous allons décrire les différents types de *paramètres* de l'*Objet-**CoCiX*** :

— Les Paramètres "*d'identité*" : Ce sont des *attributs* simples, comme le nom, date de naissance, positionnement etc...(page 8)

— Les Paramètres "*d'état*", comme la santé, la température etc...Ces paramètres fluctuent tout le long de la vie de la **CoCiX** . Ils sont issus d'un patrimoine génétique et influencés par les gènes ainsi que par son activité ou le monde extérieur. Ils sont représentés par des *Objets*.(chap 2.1.2 9)

— les Paramètres "*génétiques*" : comme le sexe, la couleur des yeux, l'agressivité, etc. Les Paramètres génétiques ne sont pas modifiables et sont acquis à la procréation par l'hérédité, ou par l'utilisateur lors de la création d'une **CoCiX** . Ils sont représentés par un *vector* d'*Objet* **Genes**.(page 17).

— Le Paramètre Cortex, est un Objet simulant les décisions que va prendre le **CoCiX** . Il y a :
  — Le Cortex Inférieur : qui s'occupe des besoins vitaux.

  — le Cortex Supérieur : qui va lui s'occuper des désirs du **CoCiX** . (Agresser, soigner etc ...)

— Les Paramètres *secondaires*, qui sont en général des *attributs* lié à des actions ou des états spécifiques du **CoCiX** .(partenaire, **.id_oeuf**, etc...)(page 19)

— Le Paramètre **desire** qui est un *Objet* dérivé de la Classe **Action** représentant le désire de la **CoCiX** .Il est déterminé par le *Cortex d'Etat*.

— Les Paramètres *d'Action*, sont des *Objets* dérivés de la Classe **Actions**.(page 20)

Nous verrons ensuite les *balises d'alerte*, qui sont des variables booléennes, informant le **CoCiX** d'un état nécessitant une action (Faim, Soif, Froid etc...)(page 20). Ils sont mis en

structure.
Il reste ensuite un tableau des 30 ancêtres du **C**o**C**i**X** (en faite il s'agit des 30 numéro d'identification **.id**)

2.1.1 Paramètres *d'identité* des CoCiX

L'identité d'un **CoCiX** est un certain nombre d'éléments pouvant l'identifier ou de la rechercher. Ils sont généralement attribués à la création de *l'Objet-CoCiX* et ne peuvent pas être modifiés, à l'exception du nom (voir plus ci-dessous).

Après le nom de l'élément, vous avez, entre parenthèse, le nom de *l'attribut* correspondante pour *l'Objet-CoCiX* :

Numéro d'identification (.id) : C'est un numéro unique, crée au moment d'une procréation ou lorsque qu'un utilisateur crée un **CoCiX** . Il est donné par le programme lors de l'ajout de *l'objet-CoCiX* dans la base de donnée.

Nom (.nom) : L'utilisateur peut mettre un petit nom à son **CoCiX** . Par défaut le nom est initialisé avec son numéro ID (voir au-dessus) précédé de la lettre majuscule F pour les femelles ou M pour les mâles.
exemple : F4335

Fichier de sauvegarde (.fichier) : Chaque **CoCiX** est stocké sur le disque dur dans le répertoire indiqué par la *variable Globale* **REPERTOIRE_NID**, dans un fichier binaire dont le nom est dans l'attribut .fichier. Le nom du fichier est de la forme XXX.ext, avec XXX son numéro d'identification (**id**). Pour un **CoCiX** vivant et né, l'extension du fichier est **.cox**, pour un **CoCiX** encore au stade d'oeuf, **.oeuf**.

Filiation (.idPere,.idMere) : C'est deux nombres sont les IDs du père et de la mère du **CoCiX** . Si elle a été crée par l'utilisateur, c'est deux variables sont mises à 0, pour indiquer qu'il n'y a pas d'ascendant.

Date de naissance (.date_naissance) : Indique la date de la procréation ou de la création si c'est un utilisateur qui a crée le **CoCiX** . La variable est un entier et correspond au jour donné par la fonction *Cycle.jour*.

Date de mort (.date_mort) : Au jour de la mort d'un **CoCiX** , **.date_mort** est initialisé avec le jour renvoyé par la fonction Cycle.jour.

Sexe (.sexe) : Détermine le sexe du **CoCiX** . Il sera matérialisé par une variable Booléenne (**False** pour Femelle et **True** pour Mâle).

Localisation (.case_presence et .case_naissance) : Le **CoCiX** se déplace dans un *Monde* fermé à 2 dimensions (voir chapitre 7 page 44).

Pour connaître l'emplacement d'un **CoCiX** , nous avons un *identifiant de case* **.case_presence**(voir chapitre 7.1 page 44).

Il y a aussi un autre *identifiant de case*, **.case_naissance**, indiquant l'emplacement de sa naissance (ou **Nid**). Il sera pour elle, un endroit de repos, de soins etc...

Vieillesse (.vieux) : Nombre de jour à partir de la procréation, qui détermine l'âge "*Vieux*" d'un **CoCiX** . Il est déterminé à la naissance, par le gène **gene[vieux]**, (voir page 33).

Un **CoCiX** rentre dans l'âge "vieux" en général vers le 20^{eme} jours.

Quand il devient "*vieux*", le **CoCiX** décline en santé (voir chapitre 2.1.2.2 page 12).
Une *méthode* existe pour connaître l'étape d'un **CoCiX** :

```
short Cocix::etape(){
        short age;
        age = (int) this->age();
        if(age <= 1) return ETAT_OEUF;
        if(age > 1 && age < MATURITE) return ETAT_BEBE;
        if(age >= MATURITE && age < vieux) return ETAT_ADULTE;
        return ETAT_VIEUX;
}
```

La méthode renvoi un short. Les variables globales permettent de connaitre la signification :
- ETAT_OEUF : Le **CoCiX** est un oeuf.
- ETAT_BEBE : Le **CoCiX** est un bébé (- de 4 jours).
- ETAT_ADULTE : Le **CoCiX** est adulte.
- ETAT_VIEUX : Le **CoCiX** est vieux.

2.1.2 Paramètres *d'état* des CoCiX

Les paramètres d'état sont des *Objet* de la classe Param_Etat informant de tous les signes vitaux des **CoCiX** . Ils sont au nombre de 4.

Vous avez :

1. Santé (**.Sante** p10)
2. Calorique (**.Calorie** p13)
3. Hydrique (**.Hydro** p15)
4. Température (**.Temperature** p16)

2.1.2.1 Propriétés des Param_Etat

Les 4 *Paramètres d'Etat* ont tous des *attributs* communs qui sont :

- L'attribut **nom** : "*Santé*", "*Température*", "*Calorie*" ou "*Hydro*".

- Des bornes supérieures et inférieures appelées respectivement *Plafond* et *Plancher*. Ces bornes sont infranchissables.

- Une valeur et une Capacité : La valeur est la représentation à l'instant t du paramètre du **CoCiX** (sa santé, son nombre de calorie, sa température ou sa quantité d'eau). La Capacité est la valeur initialisée à la création du **CoCiX** , en général issue d'un Gène. Cette valeur ne peut pas être dépassée sauf pour le paramètre **.Temperature**. Cette Capacité peut, dans le cas du paramètre **.Sante** diminuer au cours de sa vie.(Voir le vieillissement 2.1.2.2 p 12).

- Des limites "*maladie*" et "*coma*" : Chaque Param_Etat a deux seuils déclenchant les *balises d'alertes* **Malade** et **Coma**. Ces limites sont bornées en inférieur et-ou supérieur (exemple : Le **CoCiX** tombe malade si la température monte au dessus de 40˚C et tombe

dans le coma si elle descend en dessous de 35.8°C ou monte au dessus de 42°C)

— Des limites de *souffrance* : Les limites de *souffrance*, vont baisser directement le *Param_Etat* **.Sante**. Elles sont comme les limites "*maladie*" et "*coma*" vues au dessus, soit bornées d'un côté, soit des deux. Si les limites de *souffrance* sont atteintes, la santé sera diminuée d'un facteur de correction attribué à chaque *Param_Etat* et unique :

SOUFFRANCE_CALORIQUE : 0.1%
SOUFFRANCE_HYDRIQUE : 0.17%
SOUFFRANCE_THERMIQUE : 0.1%
SOUFFRANCE_MALADIE : 0.014% (Le **CoCiX** souffre d'être malade !).

Les 4 Paramètres d'Etat ont tous des Méthodes communes qui sont : :

— Les méthodes **get_** et **set_** sur **.valeur**, **.capacite** et **.nom** .

— La méthode **souffrance()** : Renvoie *Vrai* si des limites ont été franchies et si le *Param_Etat* produit de la souffrance. Cette méthode, appelé par le Cortex d'Etat, peut mettre à jours le *Param_Etat* **.Sante**, sur demande.

— La méthode **modif(** ***modificateur*****,** ***balises*****)** : qui est la méthode principale pour modifier la valeur du Param_Etat, car elle contrôle les bornes, et toutes les limites du paramètre et met à jour les balises d'alertes.(malade, coma)

— La méthode **affiche(** ***pourcentage*** **,** ***limite*****)** : Cette méthode affiche le *Param_Etat*. ***pourcentage*** indique si on veut le *Param_Etat* en pourcentage (**valeur** est alors représentée par rapport **capacité**). ***limite***, indique si on veut afficher toutes les limites du *Param_Etat*, ou seulement les informations de bases (**valeur** et **capacite**).

2.1.2.2 Santé

Les paramètres de la Santé

La santé des **CoCiX** est représentée par **2** nombres réels positifs :

— L'état de santé (*sante*) représente la santé du **CoCiX** à un instant T et est l'attribut **valeur** du *Param_Etat* **.Sante**. Il est récupérable par la méthode : **Sante.get_valeur(*pourcentage*)**.

— Le ***C****apital* ***S****anté* (**.cs**), attribué à la naissance, et enregistré dans l'attribut **capacité** du *Param_Etat* **.Sante**. Il est récupérable par la méthode : **Sante.get_capacite()**.

Les **CoCiX** ont, en moyenne, un ***C****apital* ***S****anté* (**.cs**) a 100, mais les facteurs génétiques et environnementaux peuvent faire fluctuer ce paramètre au fil des générations.

.sante est un nombre compris entre 0 et **.cs** (il est souvent représenté par un pourcentage, exemple : *100% de santé, 50% etc...*)

Santé à la naissance (*Capital Santé*)

À sa procréation, le **CoCiX** reçoit son ***C****apital* ***S****anté* (**cs**) par le gène **Genes["sante"]** (voir page 33).

Pour ça, le programme prend le gène santé et lui applique une correction $\mathcal{C}$, en regardant l'état de santé des parents au moment de la procréation.

cs = **Genes["sante"].valeur** x (1 + $\mathcal{C}$)

avec

$$\mathcal{C} = \frac{100(sante\_mere - cs\_mere)}{cs\_mere} + \frac{100(sante\_pere - cs\_pere)}{cs\_pere}$$

exemple :

Au moment de la procréation :
— Le père a **.sante** = 70 sur un Capital santé **.cs** = 100,
— La mère a **.sante** = 93 sur **.cs** = 93
— Le gène santé de l'œuf **Genes["sante"].valeur** = 95.

On a alors le Facteur de Correction (voir table 2.1) :

$$\mathcal{C} = \frac{100(93 - 93)}{93} + \frac{100(70 - 100)}{100} = -30\%$$

Soit un Capital Santé :

$$.cs = 95(1 - \frac{30}{100}) = 66,5$$

sante en %	$\mathcal{C}$
100%	+0%
90%	-10%
80%	-20%
70%	-30%
60%	-40%
50%	-50%

TABLE 2.1 – Facteur de correction de Santé du Parent

Si le **.cs** calculé est inférieur à 30, l'œuf n'est pas assez fort et meurt dans les 3 jours.
Une fois le *Capital Santé* calculé, le programme initialise le paramètre **.sante = .cs**

Le ***C**apital **S**anté* diminue avec le temps (voir chapitre 2.1.2.2 page 12).

Santé de l'œuf :

Pendant 1 à 3 jours, le future **CoCiX** est un œuf. C'est surtout la température ambiante qui va déterminer si l'œuf peut survivre. Elle doit être comprise entre $10^{\circ}C$ et $35^{\circ}C$. Sinon l'œuf meurt.

Santé du bébé :
TODO

Santé de la **CoCiX** adulte :

Voici le tableau récapitulant l'état de santé de la**CoCiX** en fonction du paramètre **sante** :

sante	**Santé de l'Adulte**
100%	Vie normale
< *seuil maladie*[1]	Malade et Souffrances
< *seuil coma*[2]	coma et Souffrances
0%	**CoCiX** mort !

La *santé* peut être diminuée par la faim (voir chap 2.1.2.3 page 13), une température ambiante trop froide ou trop chaude, les accidents, des combats, des maladies (voir chapitre 2.1.2.2 page 12) etc...

La santé peut-être récupérée par du repos, et les soins. (dans la limite de la **.cs** de la CoCiX).) voir page 13

Vieillissement

Lorsque le **CoCiX** devient "*vieux*" (voir chapitre 2.1.1 page 8), son ***C**apital **S**anté* (**.cs**) diminue d'une certaine quantité par jours marqué par le gène ['vieillissement'].
Cette quantité se calcule de deux façons différentes en fonction de **.cs** :

- **.cs >= gene['seuil_accel_vieille]** : la quantité diminue du % de vieillissement "génétique"(environ 10% par jour).
- **.cs < gene['seuil_accel_vieille]** : c'est la quantité marquée par le gène ['vieillissement'] qui est retranchée (et non un %) du ***C**apital **S**anté*(Il y a donc une accélération du vieillissement en dessous d'un certain **.cs**)

Le paramètre **sante**,lui, ne change pas mais ne doit pas dépasser le ***C**apital **S**anté*.

exemple :

- Avant le 20^{eme} jour, le **CoCiX** avait un Capital santé égal à 60 (**.cs** = 60), une santé à 50 (**.sante** = 50), il avait donc 83,33% de santé.

- Le 21*eme* jour, sa capacité de santé a diminué de 10% et est donc à 54. Sa santé, elle, reste à 50, soit 92,59%. Il semble allez mieux ! !, mais ne pourra jamais plus aller au delà de 54, comme valeur.

Maladie

Quand le **CoCiX** est *malade*, c'est en faite une souffrance d'un ou des *Param_Etat*. Les taux indiqués page 10 diminuent la santé (**.sante**). A noter qu'un **CoCiX** peut donc souffrir de plusieurs maux, température, manque calorique ou hydrique, ou simplement d'être malade (voir 2.1.2.1).

1. Le *seuil maladie* est un nombre calculé avec le gène ['*seuil_malade*'] et la **.cs** du **CoCiX**
2. Le *seuil coma* est un nombre calculé avec le gène ['*seuil_coma*'] et la **.cs** du **CoCiX**

Repos et soins :

Les **CoCiX** dorment lorsque c'est le cycle *nuit* (voir chapitre 2.2.3 page 21).

Elles *récupèrent* d'un certain pourcentage, donné par le gène **["recup_sommeil"]** (voir page 33). Ce pourcentage est d'environ **30%** de **.sante** pour 10 heures de repos.

Les **CoCiX** ont aussi la possibilité de se soigner si elles sont malades. Elles peuvent alors se régénérer d'un certain nombre de points de santé déterminé par le gène **['soins']** (ce nombre est en *.santé* par minute.). Elle va aussi réguler sa température pendant le sommeil pour l'équilibrer dans ses limites de non souffrance.

Les **CoCiX** femelle on un gène **['compassion']**, qui pourra, dans certainement condition, donner le désir de soigner ses congénères. Elle régénérera alors la santé d'un ou d'une autre **CoCiX** d'un certain nombre de points de santé déterminé par le gène **['soins']** (ce nombre est en *.sante* par minute.)

Vivante ou Morte :

Le paramètre **vivant** est une balise indiquant si la**CoCiX** est vivante ou morte. Pour **vivant = True**, alors elle est vivante, si **vivant = False** alors elle est morte !(voir chap 2.1.9 p 20)

2.1.2.3 État Calorique

Les paramètres de l'état calorique :

À la procréation, chaque **CoCiX** , reçoit un ***C**apital **C**alorique*(**.cc**) par le gène **["calorie"]** (voir page 33).

Le programme, à la naissance du **CoCiX** , initialise le paramètre **.calorie** en lui donnant la valeur du ***C**apital **C**alorique* (**.cc**)

Ces deux variables sont stockées dans le *Param_Etat* **.Calorie**. On accède à **CC** par la méthode **.Calorie.get_capacite()** et **.calorie** par **.Calorie.get_valeur()**.

— Le *Param_Etat* **Calorie** est bornée entre $[0; CC]$ en calories.

— Il n'y a pas de limites "*maladie*" pour ce *Param_Etat*.

— La limite "coma" est $[5\%deCC, \infty[$.

— La limite "souffrance" est [Gene['souffre_faim'].$valeur * CC, \infty[$.

Ce paramètre (**.calorie**) baisse pour chaque action du **CoCiX** :

Action	**Dépenses Caloriques**
Sommeil	0,016
Boire	0,3
Manger	0,5
Déplacement	1
Course	1,5
Reproduction	1,8
Combat	2

Les *dépenses caloriques* sont en **Calories** par Minute.

A noter que les femelles qui sont fécondées, ont une augmentation de leurs dépenses caloriques, de 10%.(voir chap 2.2.2)

Faim :

À partir d'un pourcentage donné par le gène **Genes["faim"]**, la**CoCiX** va ressentir la Faim et devra se nourrir.

La variable **Genes["faim"].valeur** est un pourcentage indiquant le seuil de ressenti de la faim (voir page 33).
La balise d'alerte **faim** est mise à ***Vrai*** (Voir chap 2.1.9 page 20).

Exemple :

La **CoCiX** *Toto* a un *Capital Calorique* à la naissance de **54** calories (**Gene["calorie"].valeur** = 54 => **.cc** = 54) et un gène de la faim égal à **65%** (**Genes["faim"].valeur** = 65) ;

Après avoir marché quelques heures, à la recherche de femelles, elle a un paramètre **.calorie** = 30. Elle ressent donc la faim puisque **30** est inférieur à **65**% de **54**.

Son **Genes["souffre_faim"].valeur** étant égal à 50%, la**CoCiX** va commencer à souffrir de la faim à partir de la moitié de **Genes["faim"].valeur**, soit $\frac{65}{2} = 32,5\%$ de son **C**apital **C**alorique. Si elle ne trouve pas à manger avant que **.calorie** passe sous la barre des 18 calories, elle va perdre **6%** de santé (**.sante**) par heure !

Si **calorie** tombe à 2, la **CoCiX** rentre dans le coma ! (Elle a de forte chance de mourir sauf si une autre **CoCiX** la soigne ! !).

Nourriture et Satiété :

Lorsqu'une **CoCiX** ressent la faim, elle commence à chercher de la nourriture dans son *environnement* (voir chap II page 41).
Une fois trouvé, elle puisera un certain nombre de calories, donné par le gène **Genes["assimilation_calorique"]** (voir page 33). Ces calories, récoltés s'additionnent au paramètre **.calorie**.

La **CoCiX** reste sur la zone "*de repas*",jusqu'à ce que son *seuil de satiété* soit atteint, ou qu'il n'y ai plus de nourriture.
Le *seuil de satiété* est déterminé par le gène **Genes["satiete"]** qui est un pourcentage du *Capital Calorique*(**.cc**).

Exemple :

Si le *seuil de satiété* de notre **CoCiX** *Toto*, qui vient de trouver de la nourriture, est de **95%**, ça veut dire qu'elle mangera jusqu'à ce qu'elle ait atteint un paramètre calorie = 51 (95% de 54).

2.1.2.4 État Hydrique

L'eau est un élément vital pour la vie des **CoCiX** .

Hydrométrie et l'eau :

A la procréation, chaque **CoCiX** , reçoit un *Capital Hydrique* (**.ch**), par le Gene *influent* **Genes["hydro"]** (voir page 33). Ce Capital Hydrique est en μ Litre.

Le programme, à la naissance de la **CoCiX** , initialise le paramètre Hydro (**.hydro**) en lui donnant la valeur du *Capital Hydrique* (**.ch**).

Ces deux variables sont stockées dans le *Param_Etat* **.Hydro**. On accède à **CH** par la méthode **.Hydro.get_capacite()** et **.hydro** par **.Hydro.get_valeur()**.

— Le *Param_Etat* **Hydro** est bornée entre $[0; CH]$ en calories.

— Il n'y a pas de limites "*maladie*" pour ce *Param_Etat*.

— La limite "coma" est $[8\% deCH, \infty[$.

— La limite "souffrance" est [Gene['souffre_soif'].$valeur * CC, \infty$[.

Ce Paramètre (**.hydro**) baisse pour chaque action de la **CoCiX** :

Action	**Eau dépensée en μL**
Repos & Sommeil	0,01
Repas	0,5
Déplacement	1
Course, Reproduction et Combat	3

Les *dépenses hydrique* sont par μL/Mn à une température de 37,5°C .

Un autre paramètre va jouer sur la déshydratation des **CoCiX** , c'est leur température (Voir Chapitre 2.1.2.5 page 16) :
Il y a un facteur de correction $\mathcal{C}$, qui se calcul par la formule :

$$(C) = (\tau - \tau_m)^2 \sqrt{\tau}$$

Avec τ la température *instantanée*, et τ_m la température *Normale* de la **CoCiX** , qui est de 37,5 °C.

Soif :

A partir d'un pourcentage donné par le Gene **Genes["soif"]**, la **CoCiX** ressent la Soif et doit boire.
La balise d'alerte **.soif** est mise à ***Vrai*** (Voir chap 2.1.9 page 20).
La variable **Genes.["soif"].valeur** est un pourcentage indiquant le seuil de ressenti de la soif (voir page 33).

Si **.hydro** descend en dessous d'un % du seuil de ressenti, donné par le Gene **Genes.["souffre_soif"]**, la santé (**.sante**) diminue de 10,2% par heure.

Si **.hydro** passe sous la barre des 8% du ***Capital Hydrique*** (**ch**), la **CoCiX** rentre dans le *comas* et la balise **.coma** passe à ***VRAI***.

2.1.2.5 Température

La température corporelle est un paramètre évoluant avec l'environnement, les maladies etc...
Il est fournit par le *Param_Etat* **.Temperature**, réel positif.
Les **CoCiX** ne ressentent pas la chaleur mais le froid.

À la naissance

A la procréation, le paramètre **.temperature** est initialisé à 37,5°C.

Température et Santé

La température normale d'une **CoCiX** est comprise entre 36,8°C et 37,9°C. Au delà de cette fourchette, la santé s'altère de 6% par heure.
A partir de 38°C, la **CoCiX** va avoir soif. La balise d'alerte **.soif** est mise à ***Vrai*** (Voir chap 2.1.9 page 20).
A partir de 40°C, ou en dessous de 36,8°C, la balise **.malade** est mise à ***Vrai*** (Voir chap 2.1.9 page 20).
En dessous de 35.8°C ou au dessus de 42°C la **CoCiX** tombe dans le coma.

Température et environnement :

La température corporelle des **CoCiX** va varier en fonction de la température de leur environnement :

- Si la température de l'endroit où elles se trouvent est supérieur à leur *température corporelle* (**.temperature**), alors celle-ci va augmenter de **Genes["temp"].valeur** par minute (voir chap II page 41).

- Si la température de l'environnement descend en dessous de 10°C, la température Corporelle (**.temperature**) de la **CoCiX** va descendre de **Genes["temp"].valeur** par minute.

Froid :

Le froid est ressenti par la **CoCiX** quand la température de son environnement descend en dessous d'un seuil déterminé par le Gene influent **gene.froid** (voir chap II page 41).

Ce Gene est un pourcentage de la température corporelle, servant à calculer la température limite de ressenti du froid.
La balise **.froid** sera mise à VRAI (Voir chap 2.1.9 page 20).

Une **CoCiX** qui a froid, va vouloir regagner son Nid (voir chapitre 2.1.1 page 8).

Chaleur :

Les **CoCiX** ne ressentent pas la chaleur. Leur température corporelle va augmenter et elle vont dépenser plus d'eau. c'est la soif qui les sauvera car en buvant elle vont perdre de la température à hauteur de **0,01˚C** pour **1** μ**L** d'eau pris.(voir chap 2.1.2.4 page 15)

Température et Actions :

Les différentes actions de la **CoCiX** vont faire varier sa température :

Action	**Variation Température**
Repos & Sommeil	↘ 0,05˚C/ mn
Déplacement	↗ 0,01˚C / mn
Course, Reproduction et Combat	↗ 0,1˚C/mn

En dormant la température des **CoCiX** ne peut descendre en dessous de 36,5˚C.

2.1.3 Paramètres *génétiques* des CoCiX

Les paramètres génétiques sont représentés par un *dictionnaire* (**Genome)**) d'*Objets* **gene** regroupant les caractéristiques génétiques du **CoCiX** comme la force, son agressivité ou la couleur de ses yeux ! sa capacité énergétique etc ...

Dans l'objet **CoCiX** , nous accédons à un gène spécifique par :

.Genome["*nom du gène*"]

exemple :
Pour avoir la valeur du gène *hydro* :

.Genome["hydro"].value.

Pour connaître le mode de transmission du gène *agressivite* :

.Genome["agressivite"].transmission.

Un gène peut être un *Pourcentage* ou *une valeur* :

2.1.3.1 les gènes valeur

Certain paramètres du **CoCiX** sont *influencés* par un gène. On dit que Le gène est *Influent* de la caractéristique. Ce gène va donner la valeur à la caractéristique.

exemple : Le *Capital Calorique* à la naissance (**cc**) est influencé par le gène **Genome["calorie"]** qui est transmis par le père ou la mère (voir chap 2.1.2.3 page 13).

2.1.3.2 les gènes 'pourcentage'

Certains gènes ne sont pas *influents*, mais déterminent un pourcentage ou une quantité servant à une fonction :

exemple :

Le gène **Genome.["faim"]** est un pourcentage qui détermine à partir de combien de calories dépensées le **CoCiX** ressent la faim (voir chapitre 2.1.2.3 page 14).

Les gènes sont transmis par l'hérédité.

Voici la liste des attributs de l'objet **Genome** :

Attribut	commentaire	exemple
.nom	nom du gène	Gene["hydro"].nom = hydro
.id	Numéro d'identification	Genes["hydro"].id = 5
.valeur	Valeur du gène [1]	Genes["hydro"].valeur = 100 μL
.pourc	Vrai si *Gène Pourcentage*, Faux si *Gène valeur*	Gene["soins"].pourc = false
.mod_trans	Mode de transmission [2]	(m6,pm,GP ...)
.pere	Valeur du gène, transmis par le père. [3] Il sert pour l'hérédité de ses descendants.	Genes["agressivite"].pere = 33 %
.mere	Valeur du gène, transmis par la mère. [3]	Genes["endurance"].mere = 65%
.gp_mere	Valeur du gène, transmis par le Grand-Père maternel. [3]	Genes["force"].gpMere = 65
.gp_pere	Valeur du gène, transmis par le Grand-Père paternel. [3]	Genes["fertilite"].gpPere = 80%
.gm_mere	Valeur du gène, transmis par la Grand-Mère maternel. [3]	Genes["endurance"].gmMere = 60%
.gm_pere	Valeur du gène, transmis par la Grand-Mère paternel. [3]	Genes["calorie"].gmPere = 75 cal

L'objet **Genes** est déclaré par une classe à part. (voir chapitre 3.2 page 33)
Les Gènes sont regroupés, pour un **CoCiX** , dans un *Vector*.

1. La valeur est un pourcentage si le gène est un *Gène pourcentage*.
2. Voir 2.1.3.3 p 19
3. Ces gènes peuvent ne pas du tout influencer la vie du **CoCiX** mais servent pour le transfert génétique aux générations futures.

2.1.3.3 Mode de transmission

Chaque Gène a un mode particulier de transmission donné par l'attribut **.mod_trans**. :
— P : Gène transmis par le Père.
— M : Gène transmis par la Mère.
— PM : Gène transmis soit par le Père, soit par la Mère.
— pm : Gène transmis du père au fils ou de la mère à la fille.
— GP : Gène transmis par un des deux grand-pères (saute une génération)
— GM : Gène transmis par une des deux grand-mères (saute une génération)
— H : Gène transmis que par les mâles.(Père, ou grand-pères)
— F : Gène transmis que par les femelles (Mère, ou grand-mères)
— m6 : Gène transmis aléatoirement par un des 6 membres de la famille.

2.1.4 Agressivité

Le Gene **$gene.agressivite**, représente le pourcentage qu'a la **CoCiX** à être agressive vis à vis de ses congénères adulte et de même sexe.
Exemple : si **gene.agressivite** = 33%, toutes les minutes, il y a une chance sur trois pour que la **CoCiX** , soit agressive, vis à vis d'une autre.

2.1.5 Vivacité

Le paramètre Vivacité détermine la capacité de la **CoCiX** à réagir lors d'un combat. Plus ce paramètre est élevé, plus elle a des chances d'attaquer avant l'autre. Ce paramètre est initialisé à la procréation par le Gene **gene["vivacite"]** (voir page 33).
Il va ensuite diminué à partir du moment où elle devient vieille (voir chapitre 2.1.1 page 8).

2.1.6 Autres paramètres secondaires

Récoltes et stockage Certaines **CoCiX** peuvent un moment de leur vie avoir à récolter des ressources pour les rentrer chez elles. Elle vont alors récolter des calorie sur la case où elle se trouve et vont les ramener dans leur nid. Le paramètre **recolte** est la quantité que porte à l'instant T la **CoCiX** .

2.1.7 Paramètre *Desire*

Le paramètre **desire** est un *Objet* déterminé par le **Cortex**. (Voir chap 2.3.2 page 25)

2.1.8 Paramètres *d'Actions* des CoCiX

Les Paramètres d'action sont composés de deux catégories :
— Les *Objets* représentant les actions des **CoCiX** .(Voir chap 2.3.3 page 29)

— Les paramètres servant à suivre ces Actions.

2.1.9 Les balises d'alertes

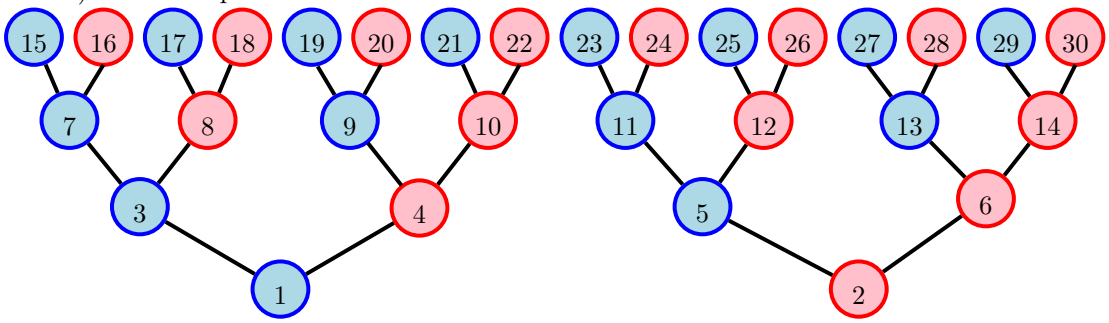
Les balises d'alertes sont des variables booléennes (binaires) qui informent la **CoCiX** d'un état (faim, soif, froid etc...) Ces balises vont entraîner des désires puis des actions spécifiques.

Le nom des balises d'alertes est assez explicite :
— **Froid**
— **Faim**
— **Soif**
— **Coma**
— **malade**
— **fecondee**

Les balises d'alertes sont mises à jour par le *cortex d'état* (voir chap 2.3.1 page 23), toutes les minutes.

2.1.10 Ancêtres

Les ancêtres du **CoCiX** sont stockés dans un tableau de 30 entiers (les Numéros d'identification **.id**). Ils sont répartis sur l'arbre suivant :



Le 1 étant le Père et 2 la mère.

Au moment du choix du partenaire, une fonction va pouvoir ainsi "remonter" 4 générations et détecter des **CoCiX** communs et ainsi calculer la consanguinité.

2 méthodes servent à récuperer ou entrer un ancêtre :
— ***.get_ancetre(numéro)*** : renvoi l'**.id** de l'ancêtre.
— ***.set_ancetre(numero, .id)*** : initialise au *numéro* d'ancêtre, le numéro **.id** de l'ancêtre.

2.2 Les étapes de la vie d'un CoCiX

2.2.1 Naissance & Mort

Un **CoCiX** a une espérance de vie de 30 à 40 jours à partir de la date de procréation.

Jour	**étape de sa vie**
0	Procréation
de 1 à 3	Oeuf
Naissance	Bébé
Naissance + 4 jours	Adulte
vieux [1]	Vieillesse

Un **CoCiX** pourra mourir de vieillesse ou prématurément de maladie ou de blessures.

A noter que l'échelle de temps d'un **CoCiX** est indiquée en *Mois*, *Jours*, *Heures* ou *Seconde*, mais que l'utilisateur pourra accélérer ou décélérer l'échelle réelle du temps. L'intérêt même de l'application, est de pouvoir étudier l'évolution génétique d'un groupe de **CoCiX** sur une très longue période.

2.2.2 Reproduction et Fécondation

Lorsque la **CoCiX** copule, le paramètre **partenaire** renseigne le numéro d'identification du partenaire sexuel.

Une foi fécondée, la balise **fecondee** est mise à *VRAI* pour la femelle.(voir 2.1.9 p20) (Cette balise est toujours à *FAUX* pour les **CoCiX** mâles !)

L'œuf crée par les deux parents est signalé par son numéro **$id** dans le paramètre **id_oeuf** de la mère.

Une fois le *cycle de ponte* passé, (voir chap 2.2.3.2 page 22) **fecondee** repasse à *FAUX*.

Voir aussi chapitre 4 page 34.

2.2.3 Les Cycles

Au cours de sa vie, les **CoCiX** vont avoir plusieurs cycles, Jour-Nuit, cycle sexuel etc ...

2.2.3.1 Cycle Jour/Nuit

Le cycle Jour-Nuit se fait sur une période de 24h, divisée en deux parties plus ou moins égales :

- **Le Jour** : Les **CoCiX** ont une activité normale, peuvent manger, boire, bouger, combattre ou se reproduire etc ...
- **La nuit** : Les **CoCiX** ne peuvent ni manger, ni se reproduire, ni combattre. Elles peuvent bouger en cas de danger et dormir.

1. **vieux** est un paramètre *d'identité* exprimé en jours depuis la date de procréation (voir chapitre 2.1.1 page 8).

Les **CoCiX** n'ont pas de notion de temps mais subissent le cycle jour-nuit, par l'intermédiaire de leur Cortex d'États (2.3.1 page 23) Une méthode de l'objet Cycle permet de connaître le cycle jour-nuit :

```
Methode jour_nuit:
Si Cycle.jour_nuit == JOUR:
Retourne "JOUR"
SI Cycle.jour_nuit == CREPUSCULE:
Retourne "CREPUSCULE"
SINON:
Retourne "NUIT"
```

Les **CoCiX** vont avoir besoin de calories pour leurs activités. Elles en dépenseront en bougeant, combattant etc... et en récupéreront en mangeant, en dormant etc.

2.2.3.2 Cycles sexuels

Les **CoCiX** ont une vie sexuelle (voir chapitre 4 page 34). C'est même souvent leur principale préoccupation après boire et manger!!

Le **CoCiX** mâle peut se reproduire de l'âge adulte à sa mort.

Seules, les**CoCiX** femelles ont un *cycle sexuel*.

De l'âge adulte, à sa mort, la **CoCiX** femelle a un cycle de 5 jours ; Elle est fécondable les 3 premiers jours de son cycle et pond le cinquième. La gestation peut donc durée de 2 à 4 jours.

TABLE 2.2 – Cycle de la **CoCiX** femelle

Jour 1	2	3	4	5
Fécond	Fécond	Fécond	-	Ponte

Il existe une *méthode* **cycleSexuel()**, qui retourne un entier compris entre -1 et 4, avec les équivalences suivantes :

Renvoyé	Cycle de la **CoCiX**
-1	Immature
0 à 2	Fécondable
3	Non Fécondable
4	Ponte si fécondée

```
Methode cycleSexuel:
cycle = ((age - MATURE ) % 5)
Si age < MATURE: Retourne IMMATURE        (-1)
SI sexe == MALE: Retourne ACTIF           (0)
Retourne cycle                            (0-4)
```

MATURE doit être une constante définie dans un fichier de constantes. Il est égal à l'âge de maturité sexuelle des **CoCiX** , soit 4 jours.

2.3 Le Système nerveux des CoCiX

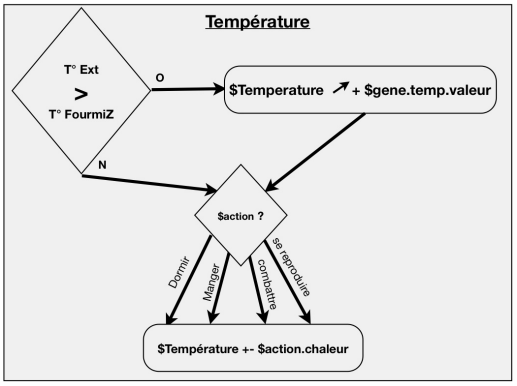
Le système nerveux des **CoCiX** est composé de quatre cortex :

1. Le Cortex d'Etat.
2. Le Cortex Inférieur.
3. Le Cortex Supérieur.
4. Le Cortex d'Action.

2.3.1 Le Cortex d'État

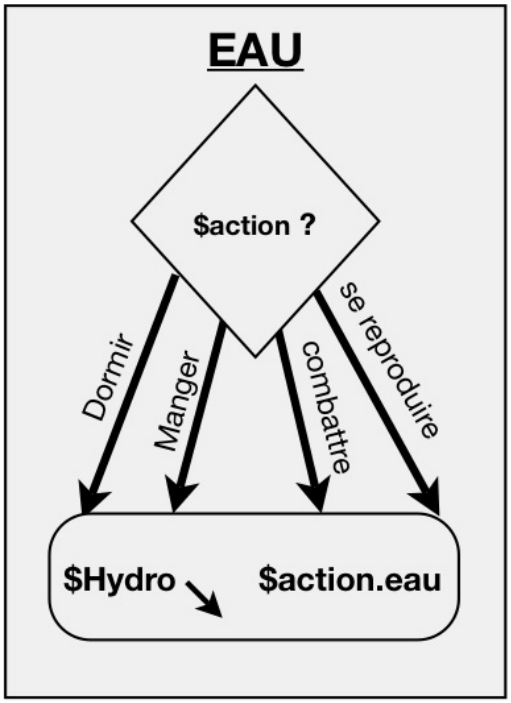
Les *paramètres d'état* vus au chapitre 2.1 sont gérés d'une façon autonome par le ***Cortex d'Etat***.
Le Cortex d'état représente le *système nerveux central* du **CoCiX** .
Il va ajuster les **paramètres** et les **balises d'état**
Le Cortex d'état est une *méthode* au sein de l'**Objet CoCiX** et qui doit être déclenché par *Cron.* [1]

1. Prend la température de son environnement et adapte la température de la **CoCiX** , en fonction de son action en cours (**action**) (voir 2.1.2.5 page 16).

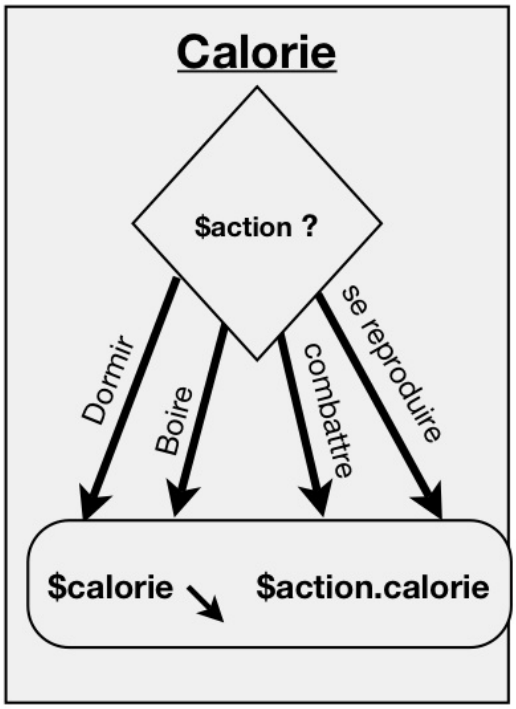


2. Dépense l'eau en fonction de l'action de la **CoCiX** .(voir 2.1.2.4 page 15)

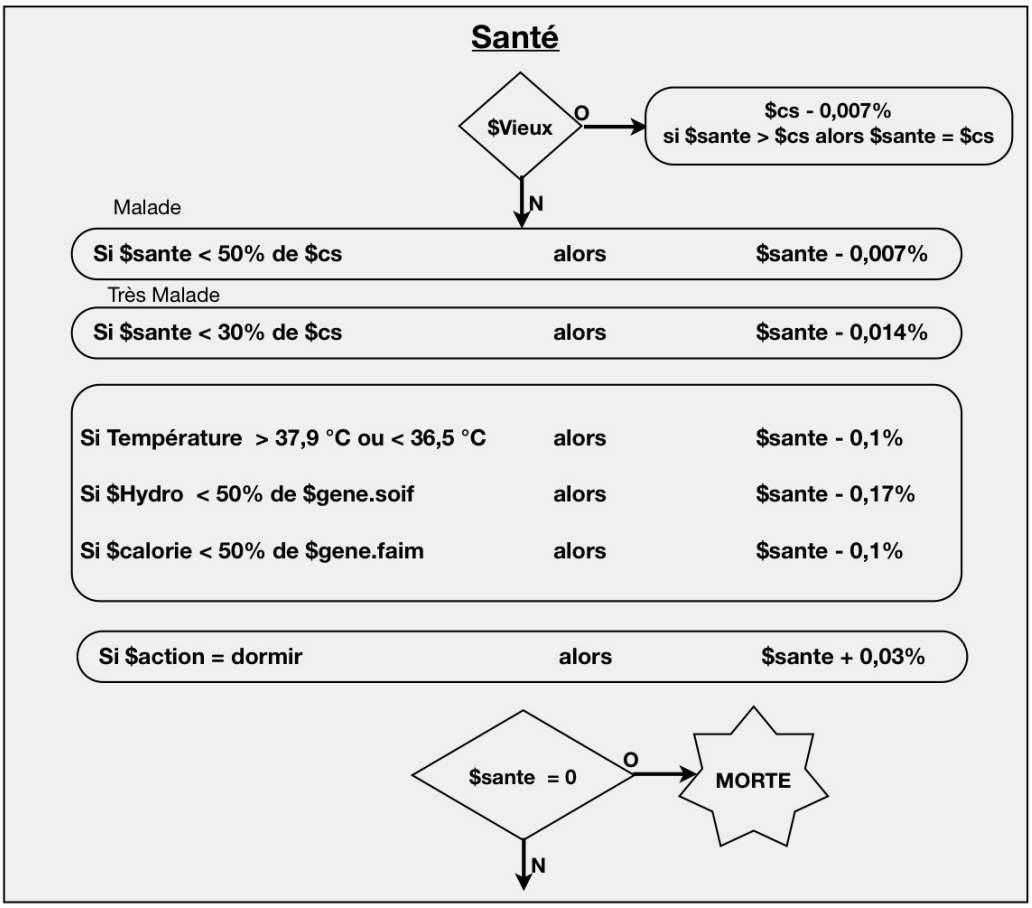
1. **cron** est un programme qui permet aux utilisateurs des systèmes Unix ou Linux d'exécuter automatiquement des scripts, des commandes ou des logiciels à une date et une heure spécifiées à l'avance, ou selon un cycle défini à l'avance.



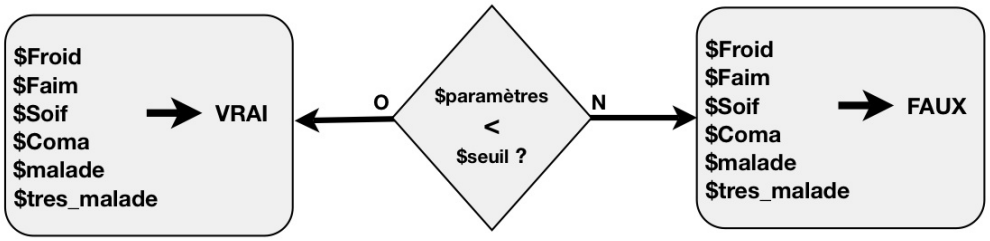
3. Dépense les calories en fonction de l'action de la **C**o**C**i**X** .(voir 2.1.2.3 page 13)



4. Calcul les *variations* de santé de la **C**o**C**i**X** .(voir 2.1.2.2 page 10)



5. Met à Jour les *balises d'alerte*.



6. Récupère le marqueur **cycle_journalier** (voir chap 6 page 41)

2.3.2 Désires

2.3.2.1 généralités

Les désires sont indiqués par le paramètre **desire**. Il est *mise à jour* par le *Cortex d'Etat* (Voir chap 2.3.1).

Il matérialise les besoins vitaux des **CoCiX** .

2.3.2.2 Les désires

Voici la liste des *Désires* possibles (verbes à l'infinitif) :

— *Agresser*
— *Boire*
— *Dormir*
— *Deposer*
— *Manger*
— *Pondre*
— *Se_chauffer*
— *Se_soigner*
— *Recolter*
— *Se_reproduire*

Ordre des décisions du **$desire** :

S'il fait jour (**jour_nuit() == JOUR**) :

1. Si **coma** est à **VRAI**, **desire** = *Dormir* et **action** = *Dort*.
2. Si **tres_malade** est à **VRAI**, **desire** = *Se_soigner*.
3. Si **Fecondee** est à **VRAI**, **desire** = *Pondre*.
4. Si **soif** est à **VRAI**, **desire** = *Boire*.
5. Si **malade** est à **VRAI**, **desire** = *Se_soigner*.
6. Si **faim** est à **VRAI**, **desire** = *Manger*.
7. Si **froid** est à **VRAI**, **desire** = *Se_chauffer*.

S'il n'y a aucune Balise d'alerte alors la **CoCiX** pourra avoir envie de :

— se reproduire si son cycle sexuel le permet : **desire**= *Se_reproduire*

— d'agresser si elle est agressive : **desire**= *Agresser*

— de récolter pour les femelles qui ne se reproduisent pas : **desire**= *Recolter* ! !

— de dormir si aucune option n'est possible : **desire**= *Dormir*

Si non :

1. Si **jour_nuit() == CREPUSCULE** Alors **desire** = Dormir, s'il n' y a aucune Balise d'alerte (faim, froid etc...)
Première heure de la nuit.

2. Si **jour_nuit() == NUIT** Alors **action** = Dort.

Une fois le paramètre **desire** positionné, c'est le *Cortex d'action* qui prend le relais pour définir l'action de la **CoCiX** (voir 2.3.3).

2.3.2.3 Structure de l'*Objet* desire

Le nom de l'*Objet* **desire** est un verbe à l'infinitif. Sa structure interne de l'*Objet* **desire** est :

— *Objet* **Action**.
— *Objet* **Alternative**.
— *Methode* **Fonction_Valide**.

Objet **Action**
L'*Objet* **Action** est un *Objet* gérant l'action permettant d'assouvir le désire (**desire**) de la **CoCiX** .

Objet **Alternative** L'*Objet* **Alternative** est un *Objet* indiquant quelle action peut être entreprise si l'action (**Action**) n'est pas possible.

Methode **Fonction_Valide** La *Méthode* **Fonction_Valide** est une *Méthode* indiquant si l'Action est faisable ou non.

Liste des *Objets* **Desire**

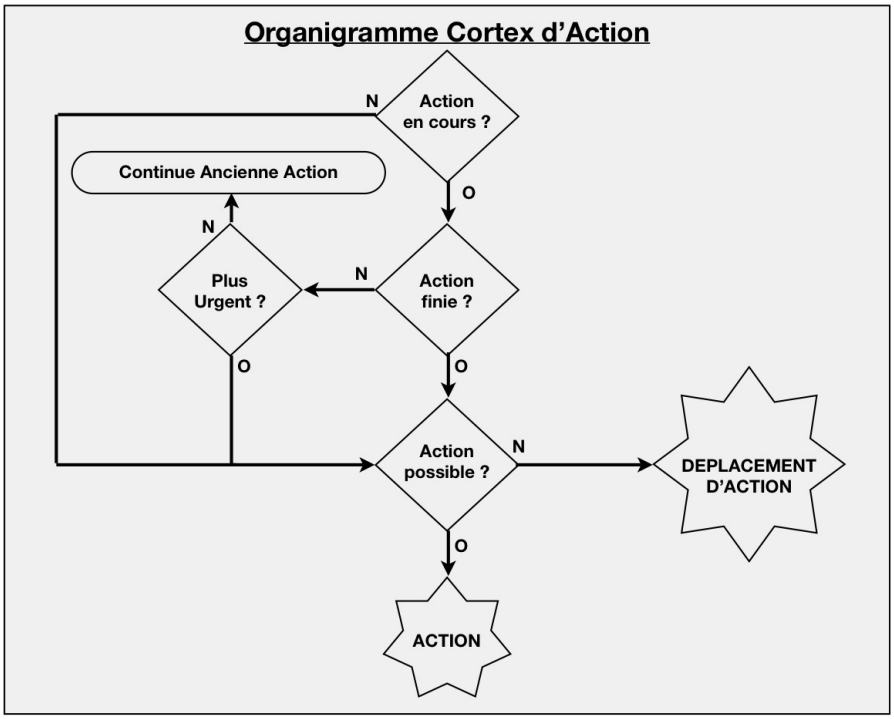
```
Manger:
        Manger.Action = Mange;
        Manger.Alternative = Cherche_Manger;
        Manger.Fonction_Valide = peut_manger();
Boire:
        Boire.Action = Boit;
        Boire.Alternative = Cherche_Eau;
        Boire.Fonction_Valide = peut_boire();
Se_Reproduire:
        Se_Reproduire.Action = Reproduit;
        Se.Reproduire.Alternative = Cherche_Partenaire;
        Se_Reproduire.Fonction_Valide = peut_se_reproduire();
Dormir:
        Dormir.Action = Dort;
        Dormir.Alternative = Rentre;
        Dormir.Fonction_Valide = peut_dormir();
Se_Soigner:
        Se_Soigner.Action = Se_Soigne;
        Se_Soigner.Alternative = Rentre;
        Se_Soigner.Fonction_Valide = peut_se_soigner();
Se_Chauffer:
        Se_Chauffer.Action = Rentre;
        Se_Chauffer.Alternative = Rentre;
        Se_Chauffer.Fonction_Valide = peut_se_rechauffer();
Pondre:
        Pondre.Action = Pond;
```

```
        Pondre.Alternative = Rentre;
        Pondre.Fonction_Valide = peut_pondre();
Recolter:
        Recolter.Action = Recolte;
        Recolter.Alternative = Cherche_Recolter;
        Recolter.Fonction_Valide = peu_recolter();
Deposer:
        Deposer.Action = Depose;
        Deposer.Alternative = Rentre;
        Deposer.Fonction_Valide = peut_deposer();
Agresser:
        Agresser.Action = Agresse;
        Agresser.Alternative = Cherche_Noise;
        Agresser.Fonction_Valide = peut_agresser();
```

2.3.3 Le Cortex d'Action

Le *Cortex d'Action* représente la partie "consciente" de la **CoCiX** .
Le *Cortex d'Action* va déterminer l'action que la **CoCiX** va pouvoir faire, en tenant compte de ses désires (**desire**), déterminés par le *Cortex d'Etat* (Voir chap 2.3.1 page 23) et de ses possibilités (terrain, partenaire, lieu etc...).

Le Cortex d'action est un programme autonome lié à chaque **CoCiX** et déclenché par cron.



Pour fonctionner, le *Cortex d'Action* va récupérer le paramètre *Objet* **Desire** transmis par le *Cortex d'Etat* (Voir 2.3.1 page 23).

exemple :

Le *Cortex d'Etat* renvoie un paramètre **Desire** = Boire :

- **Desire**.Action = Boit ;
- **Desire**.Alternative = Cherche_Eau ;
- **Desire**.Fonction_Valide = *peut_boire()*.

Le Cortex d'Action utilise des paramètres variables permettant de contrôler l'état de l'action : **temps_action** de sa partenaire.
Le paramètre d'état

- **temps_action** : indique depuis combien de temps l'action se fait (en minute)
- **Partenaire** : Partenaire si l'action implique une autre **CoCiX** *(Soins, reproduction, combat etc...)*

2.4 Les Actions

2.4.1 Principe

Action est en fait un *Objet* marquant l'action que fait la **CoCiX** .

2.4.2 Structures de l'Objet Action

Les Actions ont les *attributs* suivants :
— **.id_action** : Numéro d'identification unique (lié éventuellement à une table)
— **.action** : Verbe conjugué correspondant à l'action effectuée.
— **.chaleur** : Correspond à la variation de température qu'induit l'action (en °C par minute).
— **.eau** : Correspond à la quantité d'eau dépensée en μL par minute.
— **.calorie** : Correspond aux nombres de calories dépensées par minute pour l'action.
— **.duree** : Correspond au nombre de minutes qu'il faut pour exécuter cette action.
— **.deplacement** : Booléen indiquant si cette action est un déplacement.
— **.termine** : Booléen indiquant si cette action est terminée.
— **.fonction** : Nom de la fonction executant l'action.
— **.peut_arreter** : Booléen indiquant si l'action peut-être arrêter.

Les Actions ont les *Méthodes* suivants :
— **.go()** : Exécute l'action.

2.4.3 Liste des Actions (Verbes conjugués)

Agresse	*Fuit*
Boit	*Mange*
Cherche_eau	*Pond*
Cherche_manger	*Rechauffe*
Cherche_partenaire	*Recolte*
Combat	*Rentre*
Depose	*Reproduit*
Dors	*Soigne*

id	**action**	**desire**	**chaleur**	**eau**	**cal**	**durée**	**dep**
1	Boit	Boire	-	0	0,3	-	non
2	Cherche_eau	Boire	0,02˚C	1	1	-	oui
3	Cherche_manger	Manger	0,02˚C	1	1	-	oui
4	Cherche_partenaire	Se_reproduire	0,02˚C	1	1	-	oui
5	Combat	-	0,1˚C	3	2	-	non
6	Dort	Dormir	-0,05˚C	0,01	0,016	-	non
7	Fuit	-	0,1˚C	3	1,5	3	oui
8	Mange	Manger	-0,05˚C	0,5	0,5	5	non
9	Pond	Pondre	0,1˚C	3	3	5	non
10	Rechauffe	Se_rechauffer	-	1	3	-	non
11	Rentre	Se_soigner ou Se_rechauffer	0,01˚C	1	1	-	oui
12	Reproduit	Se_reproduire	0,1˚C	3	1,8	5	non
13	Soigne	-	0,05˚C	3	3	10	non
14	Se_soigne	Se_soigner	0,02˚C	0,1	1	5	non
15	Recolte	Recolter	0,05˚C	1	1	3	non
16	Depose	Deposer	0,02˚C	1	1	1	non
17	Cherche_recolter	Recolter	0,02˚C	1	1	-	oui
18	Agresse	Agresser	0,3˚C	5	2	5	non
19	Cherche_noise	Agresser	0,03˚C	1	1	-	oui

Chapitre 3

Le Génome

3.1 Principes et Généralités

Le génome est une base de donnée contenant tous les gènes possibles des **C**o**C**i**X** . Il sert au moment de la procréation, à initialiser tous les gènes de la nouvelle **C**o**C**i**X** .

Un gène est un *Objet* composé d'attributs :
— Un nom : *Taux_Fecondite* , *Agressivité* etc...

— Un Identifiant unique : **(ID)** Numéro donnée par la base de donnée.

— L'Influant **(influant)** : Si le gène est influent **influent** donne le nom du paramètre d'état. Il est vide si le gène n'est pas influent (voir 2.1.3 page 17).

— Un code de transmission **(transmission)** :
Ce code peut prendre comme valeur :
— P : Transmission par le père
— M : Transmission par la mère
— PM : Transmission par le père OU la Mère.
— pm : Transmission de Père en fils ou de Mère en fille.
— GP : Transmission par un des 2 Grand-pères (Saute une génération).
— GM : Transmission par une des 2 Grand-mères (Saute une génération).
— H : Transmission par les Hommes (Père, et les 2 grand-pères).
— F : Transmission par les Femmes (Mère et les deux grand-mères).
— 6p : Transmission par les 6 membres (Père, Mère, 2 grand-pères et 2 grand-mères).

Exemples : L'agressivité est un gène codé **H**, c'est à dire qu' à la procréation, l'ordinateur choisira au hasard, entre les paramètres d'agressivité (*$gene.agressivite.taux*) du Père et des deux grand-pères, et l'affectera à l'œuf.

Le *taux de fécondation* est un paramètre influencé par le gène Fécondation codé **M**, donc à la procréation, l'ordinateur affectera le même gène que la mère à sa fille.

3.2 Table du Genome

nom	Influant	Transmission	Ref
.agressivite	%	H	page 19
.assimilation_calorique	$calories	6p	page 14
.assimilation_hydrique	%	6p	page 15
.calorie	$cc	PM	page 13
.faim	$calories	6p	page 14
.froid	$temperature	H	page 16
.hydro	$hydro	GP	page 15
.recolte	$recolte	F	page 20
.recup_sommeil	$sante	M	page 10
.sante	$sante	H	page 10
.satiete	%	6p	page 14
.seuil_malade	%	6p	page 10
.seuil_tres_malade	%	6p	page 10
.soif	%	GP	page 15
.soins	%	GM	page 13
.souffre_faim	%	6p	page 14
.souffre_soif	%	6p	page 15
.temp	$temperature	PM	page 16
.vieux	$vieux	PM	page 8
.vivacite	$vivacite	GM	page 19

Chapitre 4

Sexualité & Reproduction

4.1 Principes & généralités

Lorsque deux **CoCiX** de sexe différents sont sur la même case, que le mâle est actif sexuellement et la femelle disponible et fécondable, le processus de reproduction peut commencer.
Une des **CoCiX** peut "entreprendre" l'autre en informant son **Cortex d'Action** (voir 2.3.3 page 29). Les deux **CoCiX** vont alors copuler.
A la fin de la reproduction, les gènes du père et de la mère, vont former une **CoCiX** "virtuelle" avec comme lien un numéro **Id** (voir 2.1.1 page 8).
Ce numéro id est affecté à la mère qui devient fécondée.
Lorsque la **CoCiX** fécondée passe dans son cycle de ponte (voir 2.2.3.2 page 22), elle déposera l'œuf sur la case où elle se trouve. Cet œuf aura alors comme case de naissance cet endroit.

4.2 Quand la reproduction est-elle possible ?

Pour qu'il puisse y avoir reproduction, il faut : que deux **CoCiX** de sexe différents soient sur la même case, que les deux soient matures et disponibles, et enfin que l'un des deux en exprime le désire ! !

l'action **Reproduit** est subordonnée au désire **Se_Reproduire** (voir 2.3.3 page 29). Il a comme fonction de validation : *peut_se_reproduire()*

Listing 4.1 – peut_se_reproduire()en PHP TODO

```
private function peut_se_reproduire($debug = false) {
$Partenaire = partenaire($this->case,$this,$debug);
return($Partenaire);
}
```

La fonction ***partenaire($this->case,$this,$debug)*** renvoie le numéro d'un partenaire potentiel pour la **CoCiX** placée sur la case ***case*** ou 0 s'il n'y en a pas.

S'il n'y a pas de partenaire, l'action **Reproduit** est subordonnée à *l'action alternative* **Chercher_Partenaire** (voir 2.3.3 page 29), qui ne fait que regarder autour de la **CoCiX** s'il y a des

partenaires potentiels, se déplacer vers eux ou bouger pour en trouver. Cette Fonction doit être une méthode implémentée dans l'Environnement.(Voir Chap7 p 44)

4.3 La reproduction

Lorsque tout est en place, les deux **CoCiX** vont pouvoir se reproduire.
La *Méthode* **.go()** va alors être déclenchée au niveau du *Cortex d'Action*, par l'une des **CoCiX** en question, qui en informera l'autre. La copulation a une durée d'environ 5mn et c'est le mâle qui informe la femelle du temps qu'il va mettre en mettant à jour la variable .**temps_action** de sa partenaire.
Le paramètre d'état **Partenaire** est renseigné pour chaque **CoCiX** .(voir 2.2.2 page 21)

4.4 Un ou une Nouveau(ele) CoCiX

4.4.1 principe

Lorsque la copulation est terminée, la femelle enclenche la fonction *reproduction()*qui engendrera la plupart des paramètres de la nouvelle **CoCiX** . La fonction renvoie un numéro **Id** de la nouvelle **CoCiX** , qui va être rentrée dans le paramètre **id_oeuf** de la mère.

4.4.2 Fonction *reproduction()*

Cette fonction est de la forme **reproduction(Pere,Mere)** avec **Pere** *l'Objet* **CoCiX** du père et **Mere** *l'Objet* **CoCiX** de la mère.
La fonction contient plusieurs étapes :

— *Insertion* dans la base de donnée d'une nouvelle **CoCiX** . On lui met la plupart des paramètres à 0 sauf celui des identifiants des parents : (**.idpere** et **.idmere**). On récupère alors le numéro id de cette nouvelle **CoCiX** , afin de rentrer dans la table gènes, le patrimoine génétique de l'œuf.

— *"Mixagénèse"* de la nouvelle **CoCiX** :
Cette étapes consiste à prendre les gènes, un par un dans la table du génome (voir 3 page 32), puis à leur appliquer :

1. Transmission génétique : cette étape récupère les gènes de chaque parents et les copies dans le patrimoine de l'oeuf :

```
Genes[nom du gène].pere = Pere.Genes[nom du gène].valeur
Genes[nom du gène].mere = Mere.Genes[nom du gène].valeur
Genes[nom du gène].gp_mere = Mere.Genes[nom du gène].pere
Genes[nom du gène].gp_pere = Pere.Genes[nom du gène].pere
Genes[nom du gène].gm_mere = Mere.Genes[nom du gène].mere
Genes[nom du gène].gm_pere = Pere.Genes[nom du gène].mere
```

2. Chaque gène donne son mode de transmission et l'on choisit au hasard entre les gènes transmis, la valeur à attribuer au gène de l'œuf.

   exemple : pour le gène **.soif**, transmit par GP, on va prendre au hasard la valeur du gène **.soif** des deux grand-peres de l'œuf, c'est à dire entre :
   Genes['soif'].gp_mere et
   Genes['soif'].gp_pere

   Le hasard de notre exemple veut que cela soit celui du grand-père maternelle ...
   On initialise donc la valeur de ce gène avec la valeur du grand-père maternelle :

   Genes['soif'].valeur = Genes['soif'].gp_mere

3. Modification "*consanguine*" : Tous les gènes subissent une altération dûe à la consanguinité. Le paramètre renvoyé par la fonction *degre(**CoCiX**,**CoCiX**)* est un pourcentage d'altération de + ou - cette valeur.

4. Mutation environnementale : Le facteur environnemental est tributaire du taux de radioactivité de la case où à eu lieu la copulation. La valeur du gène est affecté de +ou - 10% de la radioactivité.

   A noter que les modifications sont cumulées pour le calcul final de la valeur du gène :

   exemple : La valeur du gène **.soif** avant les modificateurs est de 60%. La fécondation a eu lieu sur une case ayant un taux de radioactivité = 40 Rad, on a alors :

   — + ou -5% de modification naturelle

   — 10% de 40 = + ou -4% de Mutation environnementale

   On a en tout 9% en plus ou en moins de modification de la valeur du Gène **.soif**, soit une valeur comprise entre **54,6%** et **65,5%** .

5. Récupération des valeurs importantes, affectant des *Paramètres d'État* (voir 2.1.2 page 9) ou génétiques (voir 2.1.3 page 17) :

   — **.vieux** : initialisé par le gène : **Genes['vieux']**. (voir 2.1.1 page 8)
   — **.ch** : Le **C**apital **H**ydrique est initialisés par le gène **Genes['hydro']**.(voir 2.1.2.4 page 15)
   — **.cc** : Le **C**apital **C**alorique est initialisé par le gène **Genes['calorie']**.(voir 2.1.2.3 page 13)
   — **.cs** : Le **C**apital **S**anté est initialisé par le gène **Genes['sante']**.(voir 2.1.2.2 page 10)

6. Enregistrement du gène de l'œuf dans la base de donnée *Gènes* :
   Chaque gènes ainsi crées à partir du patrimoine génétique de la mère et du père, est enregistré dans la base de données *Gènes* associé au numéro de la future **CoCiX** .

— On a récupéré avec la *Mixagénèse*, les paramètres états et génétiques (**.vieux**, **.ch**, **.cc** et **.cs**) qui nous permettent de mettre à jour la future **CoCiX** . Les paramètres **.case** et **.date_naissance** sont laissés à 0 tant que l'œuf n'a pas été pondu.

— La fonction retourne enfin le numéro de l'œuf à la mère.

4.5 La ponte

À la période de ponte la femelle va avoir le *désire* de pondre au niveau de son *Cortex d'Etat* :

.Desire = Pondre

L'action **Pond** est subordonnée au désire de Pondre et à une fonction de validation : *peut_pondre*().

La **CoCiX** peut pondre si :
— C'est une femelle.
— Elle est fécondée.
— C'est son cycle de ponte.
— Une place est libre sur la case.

Chapitre 5

Les Combats

5.1 Principe

Les **CoCiX** ont un gène d'agressivité (voir page 19) , et peuvent avoir le désire de combattre et de chercher des ennemis potentiels auprès de ses congénères adultes et de même sexe.
Le principe est assez similaire à la reproduction (voir chap 4 p 34).

Une des **CoCiX** peut attaquer l'autre en informant son **Cortex d'Action** (voir 2.3.3 page 29). Les deux **CoCiX** vont alors combattre.

5.2 Quand le combat est-elle possible ?

Pour qu'il puisse y avoir combat, il faut : qu'une **CoCiX** agresse une autre **CoCiX** du même sexe sur la même case.

l'action **Agresse** est subordonnée au désire **Agresser** (voir 2.3.3 page 29). Il a comme fonction de *validation* : **peut_agresser()**.

S'il n'y a pas de **CoCiX** à agresser, l'action AGRESSE est subordonnée à *l'action alternative* **Chercher_Noise()** (voir 2.3.3 page 29), qui ne fait que regarder autour de la **CoCiX** s'il y a des ennemis potentiels, se déplacer vers eux ou bouger pour en trouver.

5.3 Le Combat

Lorsque le combat commence, c'est la **CoCiX** qui a son **.id** le plus petit qui compte les tours. les *actions* se déroulent dans un ordre précis :

1. Initiative. (qui attaque en premier)
2. début du tour attaquant.
3. jet du dé d'attaque
4. jet du dé de défense

5. rapport force endurance
6. rapport de blessures
7. rapport de courage pour savoir si la **C**o**C**i**X** fuit.
8. fin du tour attaquant, début tour défenseur.
9. seq 3-6

TODO : Détailler les combats dans l'esprit *Warhammer* !

Deuxième partie

Environnement

Chapitre 6

Généralité, notion de temps et bases de données

L'environnement des **C**o**C**i**X** est un véritable petit *Monde* dans lequel, elles vont pouvoir, se déplacer, puiser des ressources (nourriture, eau etc...), faire des rencontres ! se reproduire etc... Ce monde est autonome, c'est à dire qu'un programme, va le fait évoluer, tant en ressource, qu'en météo etc...

Il a sont propre temps, gérer par des *crons*[1] du serveurs.
Plusieurs bases de données vont être nécessaires pour représenter l'environnement :

— Une base de donnée pour les *cases* du monde (voir 7.3).
— Une base de donnée des cycles (Jour-Nuit, Jours).

6.1 La Base Cycles

La base de donnée *cycles* contient plusieurs marqueurs :

1. **$bigbang**,indique la date de la création du monde.
2. **$vitesse**,indique la vitesse à laquelle passe le temps.
3. **$minute** est incrémenté chaque minute par *crons*[1]. Il augmente de 1 multiplié par le marqueur **$vitesse**. (exemple : si **$vitesse** = 4, chaque minute serveur augmente le temps du monde de 4 minutes).
4. **$heure** : indique l'heure du monde.
5. **$ jour_nuit** est un entier = JOUR, NUIT ou CREPUSCULE (constantes)
6. **$jours** indique le jour du monde. C'est aussi le nombre de jour écoulés depuis le Bigbang !
7. **$heure_nuit** donne l'heure où il fait nuit.
8. **$heure_jour** donne l'heure où il fait jour

Cette base de données est automatiquement mise à jour par *cron*[1] de serveur.

1. *cron* est le nom d'un programme qui permet aux utilisateurs des systèmes Unix d'exécuter automatiquement des scripts, des commandes ou des logiciels à une date et une heure spécifiées à l'avance, ou selon un cycle défini à l'avance.

Listing 6.1 – jour_nuit_cron_php

```
%\begin{verbatim}
//Cycle JOUR-NUIT
// par le cron
include_once("connexion.php");

define("JOUR",1);
define("NUIT",-1);
define("CREPUSCULE",0);

$sql1 = "SELECT * FROM cycles";

$resulta_requete = mysql_query($sql1);
if($resulta_requete <> false) {
$cycles = mysql_fetch_object($resulta_requete);

$vitesse = $cycles->vitesse;
$minute = $cycles->minute;
$heure = $cycles->heure;
$jours = $cycles->jours;

$heure_jour = $cycles->heure_jour;
$heure_nuit = $cycles->heure_nuit;
$jour_nuit = $cycles->jour_nuit;

// incremente le cycle journalier
$minute = $minute + ($vitesse);
if($minute >= 60) {

// une heure de passee
$minute = ($minute-60);
$heure++;
if($heure >= 24) {
// un jour de passe
$heure = $heure-24;
$jours++;
}
}
// determination de la luminosite :

if(($heure == ($heure_nuit - 2)) or ($heure == ($heure_nuit-1))) {
$jour_nuit = CREPUSCULE; // Il faut aller se coucher
} else {

if($heure >= ($heure_nuit) or $heure < $heure_jour) {

// il fait nuit
$jour_nuit = NUIT;
} else {
```

```
// il fait jour
$jour_nuit = JOUR;
}
}

echo "Jour # $jours , $heure h, $minute mn</br>";
switch($jour_nuit) {
case -1:
echo "Il fait nuit.";
break;
case 0:
echo "Il va bientot faire nuit.";
break;
case 1:
echo "Il fait jour.";
break;
}

$update = "UPDATE cycles SET minute = '".$minute."', heure = '".$heure.
"', jours = '".$jours."', jour_nuit = '".$jour_nuit."' ,date = NOW( ) WHERE id=1";

mysql_query($update);

mysql_close($liendb);

} else {
die("Impossible de charger les cycles sur le serveur...");
return false;
}

%\end{verbatim}
```

Chapitre 7

Monde

Le *Monde* est une grille de 100 x 100 cases, dans un espace fermé.
Chaque case représente un espace de $1cm^2$, pouvant contenir 2 **C**o**C**i**X** maximum.

7.1 Coordonnées (x,y)

7.1.1 Principes

Les cases sont numérotées de gauche à droite et de haut en bas.

Elles peuvent être aussi représentées par un couple (x,y) (x pour le nombre de cases Horizontales et y pour les Verticales).
La case (1,1) se situe en Haut à Gauche, et son **id** est 1.

Pour passer du numéro unique (**id**) au couple **(x,y)** on a :

x = Reste de la division entière de **id** par 100 (*Modulo*). (Si le *Modulo* est à 0 alors x=100).

Ensuite,

$$y = \frac{\mathbf{id} - x}{100} + 1 \tag{7.1}$$

exemple :

Notre **C**o**C**i**X** TOTO est sur la case dont l'**id** est 2928.

On a

$$x = Modulo(\frac{2928}{100}) = 28,$$

et

$$y = \frac{2928 - 28}{100} + 1 = 30.$$

la case de TOTO est représentée par le couple **(28,30)**.

7.2 Orientation & Déplacements

7.2.1 Principes

La **CoCiX** devra se déplacer, et donc s'orienter.

Pour se déplacer :
— De gauche à droite, il suffit d'ajouter 1 à son numéro unique de case.
— de droite à gauche, on retranche 1 au numéro unique de case.
— Pour monter il suffit de retrancher 100.
— Pour descendre d'ajouter 100.

L'espace étant fermé, le bord *Droit* et le Bord *Gauche* de la carte du monde se "*touchent*", ainsi que le bord *supérieur* et *inférieur*.
Le passage du bord droit au bord gauche, se fait automatiquement par l'addition du numéro de la case.
De même pour le passage de gauche à droite, le retranchement fait passer automatiquement d'un bord à l'autre.

Le logiciel contrôle ensuite le *numéro unique* de case qui doit être compris entre 1 et 10000, en additionnant ou soustrayant 10 000.

Exemple :
Lorsque la **CoCiX** est au bord supérieur en étant en (10,1) soit en case n°10, Si elle monte d'une case, on retranche 100 donc sa case est –90. En additionnant 10 000 on trouve 9910 qui est la case correspondante en bas. (10,100).
A l'inverse si la **CoCiX** est au bord inférieur, par exemple à case 9922 (22,100), et qu'elle descend d'une case on aura 9922 + 100 = 10 022 - 10 000 = 22 case (22,1).

7.2.2 Fonctions

Voici les 4 fonctions pour se déplacer. Le paramètre passé est le numéro de la case de départ. Les fonctions renvoient la case d'arrivée après l'avoir validée :

Listing 7.1 – Fonctions de déplacements dans 'Monde'

```
fontion monte(depart){
        arrivee = depart - 100;
        retourne valide_case(arrivee);
}

fonction descend(depart){
        arrivee = depart + 100;
        retourne valide_case(arrivee);
}

fonction gauche(depart){
        arrivee = depart - 1;
        retourne valide_case(arrivee);
}
```

```
fonction droite(depart){
        arrivee = depart + 1;
        retourne valide_case(arrivee);
}
```

La fonction valide_case() contrôle l'espace fermé :

Listing 7.2 – Fonctions dans Monde

```
fonction valide_case(case){
        Si case <= 0  retourne case+10000;
        Si case > 10000 retourne case-10000;
        Retourne case;
}
```

Nous avons ensuite les fonctions qui combinent monter, descendre, droite et gauche :

Listing 7.3 – monde.php

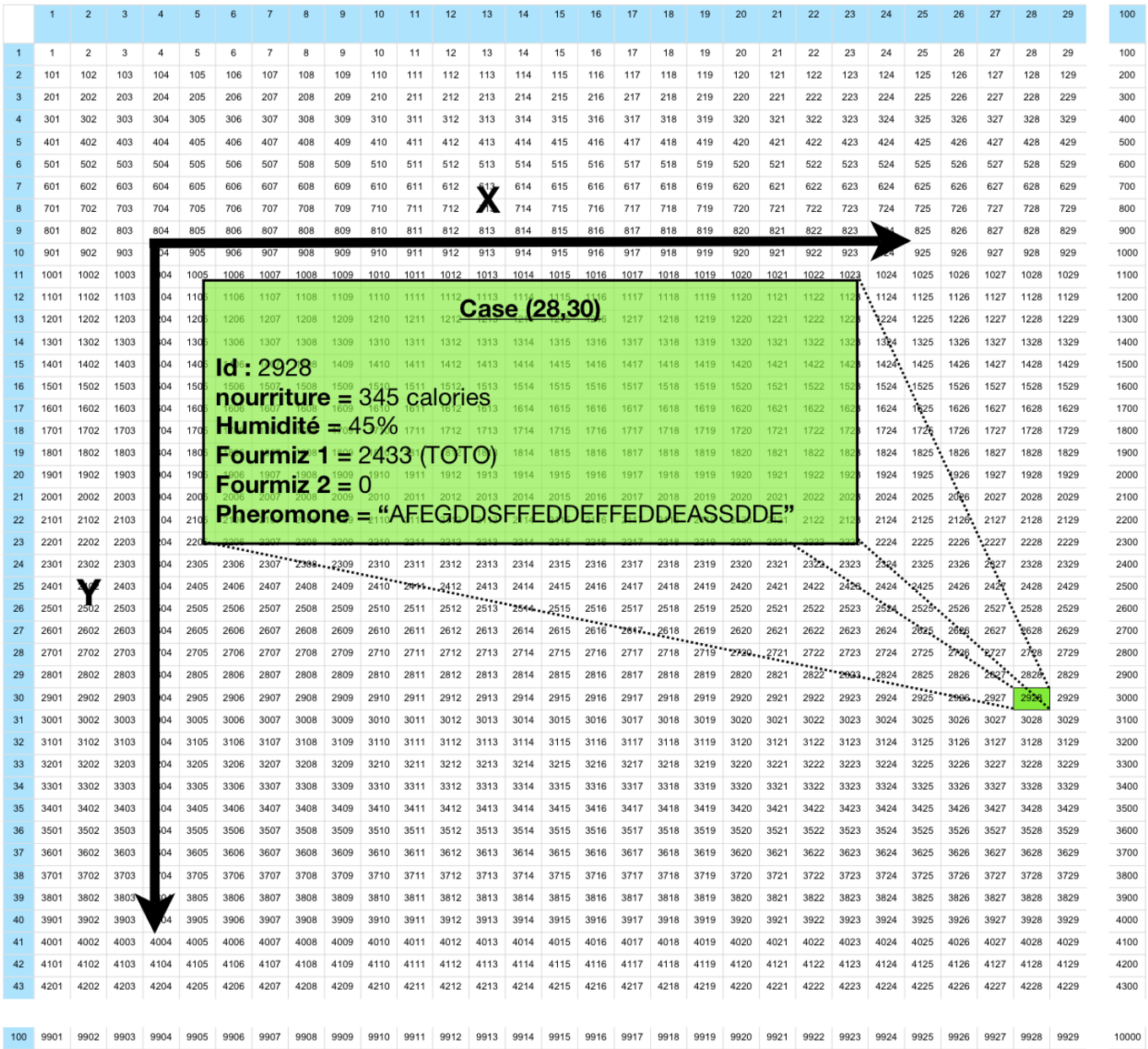
```
function monte_gauche($depart){
$etape = monte($depart);
$arrivee = gauche($etape);
}

function monte_droite($depart){
$etape = monte($depart);
$arrivee = droite($etape);

}

function descend_gauche($depart){
$etape = descend($depart);
$arrivee = gauche($etape);
}

function descend_droite($depart){
$etape = descend($depart);
$arrivee = droite($etape);
}
```

7.3 Base de données Monde

Le Monde des **CoCiX** est représenté informatiquement par une base de donnée de 100 x 100 = 10 000 cases. Chaque case contient des informations sous la forme de champs :

Id : numéro unique de la case.
nourriture : Quantité de nourriture disponible (en calories).

humidité : Quantité d'eau disponible (en μL).
temperature : Temperature de la case (en degré Celsius).
CoCiX1 : **$id** de la première **CoCiX** (à 0 s'il n'y en a pas).
CoCiX2 : **$id** de la deuxième **CoCiX** (à 0 s'il n'y en a pas). [1]
pheromone : Les **CoCiX** peuvent laisser des phéromones sous la forme d'un code (TODO).
radio : Taux de radioactivité en Rad.



1. Possibilité d'utiliser une base jumelée pour qu'il y ai autan de **CoCiX** que l'on veut sur une case.

7.4 Fonctions

Plusieurs *fonctions*, appelées par la **CoCiX** ou par des modules extérieurs, permettent de connaître les informations sur une case précise :

Ces fonctions sont pour l'instant écrite en *Php*.

7.4.1 Fonction Temperature()

Cette fonction renvoi la température d'une case passée en paramètre à cette fonction :

Listing 7.4 – monde.php

```
.../...
function temperature($case) {
// renvoi la temperature exterieur de la case
if($case > 1 && $case <= 10000) {
$sql = "SELECT temperature FROM monde WHERE id='".$case."'";
$resulta_requete = mysql_query($sql);
if($resulta_requete <> false) {
$case_monde = mysql_fetch_object($resulta_requete);

return ($case_monde->temperature);

} else {
die("Impossible de charger les cycles sur le serveur...");
return 0;
}
} else die ("erreur");

}
.../...
```

Troisième partie

Classes et Objet

Chapitre 8

Les déclarations de Classes

8.1 Généralités

Ce chapitre va rentrer dans les détails des Classes d'Objets. TODO car elles sont pour l'instant écrite en Php.

8.2 La Classe Gene

Listing 8.1 – Classe Gene

```
class Gene {
public $nom;
public $id;
public $influent;
public $valeur;
public $pere;
public $mere;
public $gp_mere;
public $gp_pere;
public $gm_mere;
public $gm_pere;
}
```

8.3 La Classe CoCiX

8.3.1 Les définitions de la Classe CoCiX

Listing 8.2 – Classe CoCiX

```
class CoCiX {
// parametres d'identite
public $id;
```

```
public $nom;
public $idpere;
public $idmere;
public $date_naissance;
public $sexe;
public $case;
public $case_naissance;
public $vieux;

// parametres genetiques
private $Genes = array();

//parametre d'etats
//sante
public $cs;
public $sante;
public $vivant;

//Energie
public $cc;
public $calorie;

//eau
public $ch;
public $hydro;

//temperature
public $temperature;

//balises d'alertes
public $froid;
public $faim;
public $soif;
public $coma;
public $malade;
public $tres_malade;

//Cortex d'etat
public $desire;

}
```

8.3.2 Les fonctions d'alertes

Les fonctions d'alertes mettent à jour les balises d'alertes (voir chapitre 2.1.9 page 20).

Listing 8.3 – Fonctions d'alertes

```
private function alerte_froid(){
//on prend la temperature exterieur de la case
$temp_exterieur = temperature($this->case);

// on calcul la temperature de ressenti du froid
$temp_froid = $this->temperature -
($this->temperature*$this->Genes["froid"]->valeur);

if($this->rentree()){
// si la CoCiX est chez elle elle n'a pas froid!
$this->froid = false;
} else {
$this->froid = ($temp_exterieur < $temp_froid);
}
}

private function alerte_faim(){
// met l'alerte Faim si le seuil de ressenti de la faim est depasse
$this->faim = (($this->calorie/$this->cc) < $this->Genes["faim"]->valeur);

// verifie que le seuil des 5% n'est pas atteint si oui -> coma
if(($this->calorie/$this->cc) <= 0.05) $this->coma = true;
}

private function alerte_soif(){
//met a jour la balise si  le seuil de ressenti de la soif est depasse
$this->soif = (($this->hydro/$this->ch) < $this->Genes["soif"]->valeur);

//verifie que le seuil des 8% n'est pas atteint si oui -> coma
if(($this->hydro/$this->ch) <= 0.08) $this->coma = true;
}

private function alerte_malade(){
$this->malade = (($this->sante/$this->cs) < 0.5); // 50%
$this->tres_malade = (($this->sante/$this->cs) < 0.3); //30%
}
```

8.3.3 chargement d'une CoCiX

Le chargement d'une CoCiX ce fait par la fonction *load($id)*, avec $id l'identifiant de la **CoCiX** dans la base de donnée.
la fonction interroge premièrement la base *CoCiX*, pour y charger les paramètres identités et d'états, puis va chercher les gènes sauvegardés dans la base *genes*, qui regroupe par numéro **ID** de **CoCiX** , tous ses gènes.

Listing 8.4 – Chargement d'une CoCiX

```
public function load($id){
```

```
// chargement d'un CoCiX exitante

$sql1 = "SELECT * FROM CoCiX WHERE id = '$id'";
$resultat_CoCiX = mysql_query($sql1);
if($resultat_CoCiX <> false) {
$CoCiX = mysql_fetch_object($resultat_CoCiX);

//parametres d'identites
$this->id = $id;
$this->nom = $CoCiX->nom;
$this->idpere = $CoCiX->idpere;
$this->idmere = $foumiz->idmere;
$this->date_naissance = $CoCiX->date_naissance;
$this->sexe = $CoCiX->sexe;
$this->case = $CoCiX->case;
$this->case_naissance = $CoCiX->case_naissance;
$this->vieux = $CoCiX->vieux;

// parametres d'etat
$this->cs = $CoCiX->cs;
$this->sante = $CoCiX->sante;
$this->cc = $CoCiX->cc;
$this->calorie = $CoCiX->calorie;
$this->ch = $CoCiX->ch;
$this->hydro = $CoCiX->hydro;
$this->temperature = $CoCiX->temperature;

// on charge les genes
$sql2 = "SELECT * FROM genes WHERE id_CoCiX = '$this->id'";
$resultat_genes_CoCiX = mysql_query($sql2);

while($genes_CoCiX = mysql_fetch_object($resultat_genes_CoCiX)){
$nom_du_gene = $genes_CoCiX->nom;
$this->gene.$nom_du_gene = new gene($genes_CoCiX->id,$genes_CoCiX->influent,$genes_Co
return true;
} else{
die("Impossible de charger la CoCiX # $this->id");
return false;
}
}
```