HMIN108 - Programmation orientée agents
Partie 1: Agents réactifs
1 - Bases et simulations multi agent

Suro François
(adaptation des cours de Jacques Ferber)

Université de Montpellier
Laboratoire d'informatique, de robotique
et de microélectronique de Montpellier

Septembre 2020

Domaine

Progression des techniques et concepts de programation

- language machine
- assembleur
- scripts
- structures de données
- objets
- agents

Différentes approches agent

- raisonement symbolique
- déductif
- raisonment pratique
- réactifs
- hybrides

Agents réactifs et simulations

Rodney Brooks: critique de l'IA symbolique

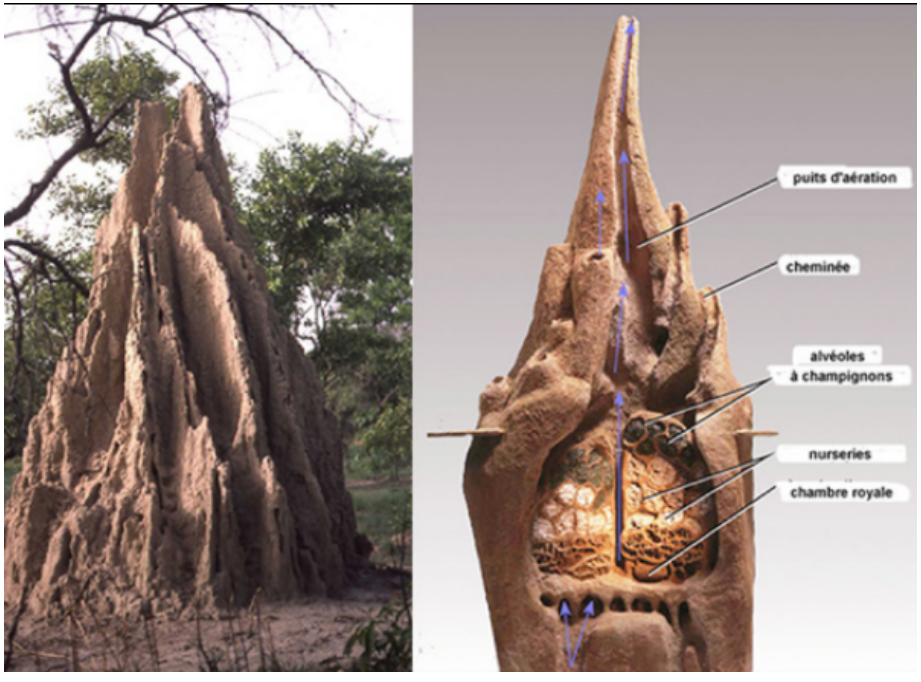
- Un comportement intelligent peut apparaitre sans représentations de type symboliques
- Un comportement intelligent peut apparaitre sans raisonment abstrait
- l'intelligence est un propriété *émergente* de certain systèmes complexes
- Une autre approche :
  - L'intelligence telle qu'on la connait est située et incarnée
  - Les comportement intelligents émergent à travers les interactions avec l'environnement

Simulations **multi** agent

- Un grand nombre d'agents simples
- Un grand nombre d'interactions avec l'environment
- Un grand nombre d'interactions **sociales**

Des sociétés animales

Métaphore sociale pour des entités autonomes simples ...



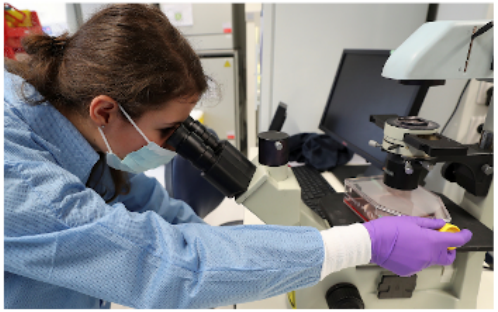
Aux sociétés humaines

.. ou complexes

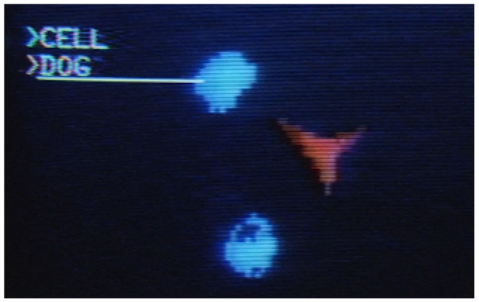
Points communs:

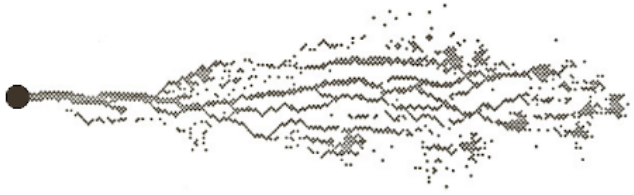
- Existence d'individus
- Ces individus interagissent dans un environnement
- Ils coordonnent leurs actions et/ou entrent en conflit/compétition
- Il existe une organisation sociale fondée sur la notion de rôle (métier, caste, poste, etc..) et de groupes (familles, nid, tribus, sociétés, etc..)
- Ils produisent collectivement des structures (objets, activités ...), qu'il n'auraient pas pu créer individuellement

Les modèles



Modéliser pour comprendre





S'inspirer pour proposer des solutions techniques

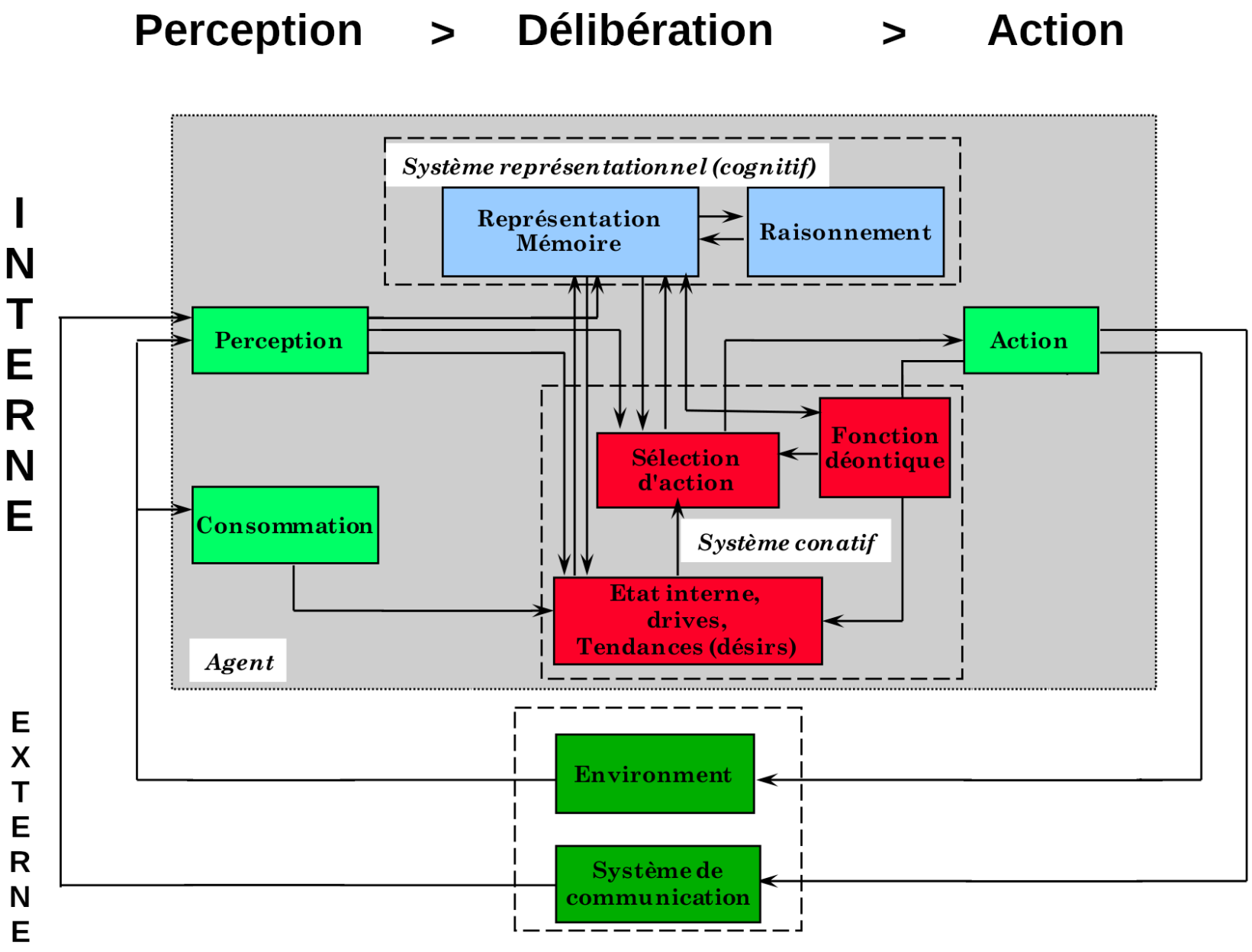
Un agent

Un **agent** est une entité physique ou logicielle **située** dans un environnement (réel ou virtuel) qui est capable de :

- **Agir** dans son environnement
- **Percevoir** et partiellement se représenter son environnement (y compris les autres agents)
- **Communiquer** avec d'autres agents
- Mû par une **volontée propre** (buts, survie, satisfaction, drives)
- Se **conserver** et se **reproduire**
- Jouer un (ou plusieurs) **rôle** dans une organisation

Un agent présente un **comportement autonome** qui est la conséquence de sa perception locale de l'environnement, de ses représentations propres et de ses interactions avec d'autres agents.

Architecture



Systeme multi agent

Un SMA est défini comme:

- Un ensemble C d'entités situées dans un environment E (E est caractérisé par un ensemble d'états possibles S)
- Un ensemble A d'agents avec A inclus dans C
- Un système d'actions permettant à des agents d'agir dans E (une action permet de passer d'un état S à un état S')
- Un system de communication entre agents (envoi de messages, diffusion de signaux, traces dans l'environement ...)
- Une organisation O structurant l'ensemble des agents et définissant leurs fonctions, éventuellement organisé en groupes ou les agents jouent un rôle.

Concepts

Pas de centralisation

- Autonomie
- Portée locale
- Contrôle distribué

Interarction entre agents

- Coordination
- Coopération
- Compétition

Système émergent

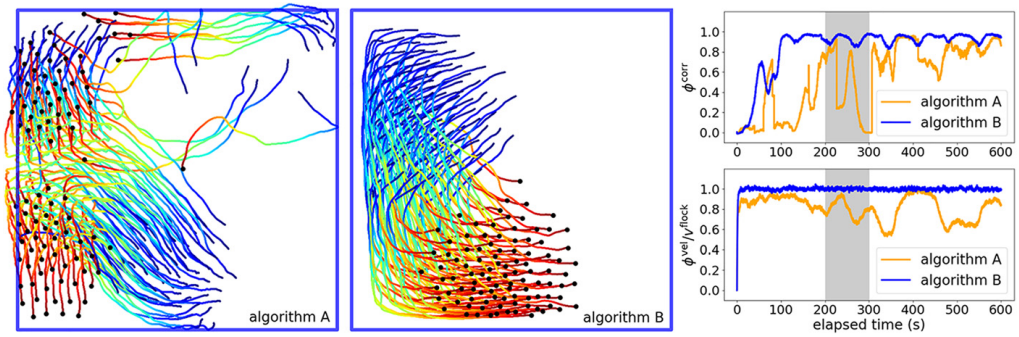
- Niveau micro: comportement de l'agent
- Niveaux macro: comportement de la société dans son ensemble

Applications

Simulations de foules pour le cinéma (scène du seigneur des anneaux utilisant le logiciel Massive)

Jeux vidéos (IA, NPC, navigation...)

Simulation industrielles et scientifique (Plans d'évacuation incendie, controle de drones...)



Niveaux de cognition

Agents "réactifs":

Qui ne disposent pas d'une représentation explicite de leur environnement

- Actions situées, pas de persistance
- EX: fourmis

Agents "cognitifs":

Qui ont une représentation de leur environnement, d'eux-mêmes et d'autres agents et qui peuvent raisonner sur leurs représentations

- Planification
- EX: humains

Agent réactif

Exemple: un garde dans un jeu video

- Tant que je ne vois rien, je suis mon chemin de garde
- Si je vois un ennemi :
  - S'il n'est pas menaçant et que je ne suis pas blessé, j'attaque !
  - S'il est menaçant ou si je suis blessé, je fuis, j'apelle à l'aide ...

Comportement émergent: les termites

Construction de tas de bois

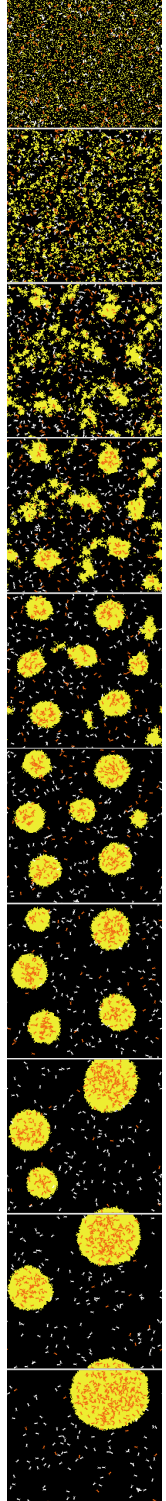
Comme les termites lors de la construction de leur nid

Termites assemblent des morceaux de bois et les empilent. Les termites suivent un ensemble de règles simples individuelles et locales

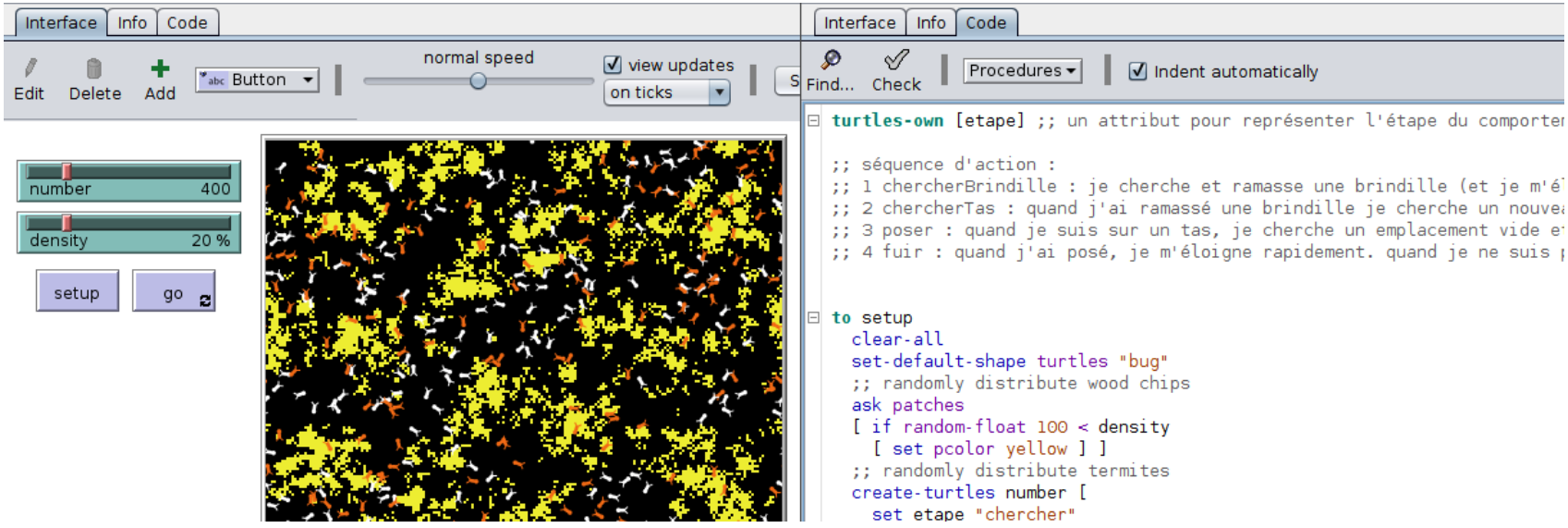
Règles

- Si je rencontre un morceau de bois, je prend le morceau et continue mon chemin.
- Quand je porte un morceau de bois et que je rencontre un autre morceau de bois par terre, je cherche un coin vide et je dépose mon morceau de bois.

Avec ces règles, finalement, les regroupements de bois, se transforment en piles..



NetLogo



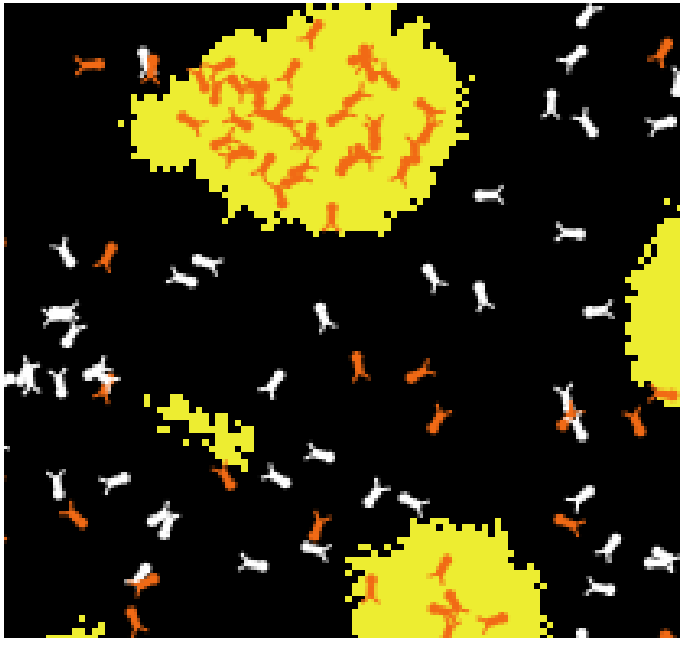
Environnement de développement multi-agents réactifs, pour l'étude de systèmes complexes

- On peut gérer des centaines (voire des milliers) d'agents qui opère en même temps dans un environnement
- Ecrit en Java
- Très facile à utiliser
- Interface conviviale..
- Tourne sur toutes les machines (Windows, Mac OS,Linux)

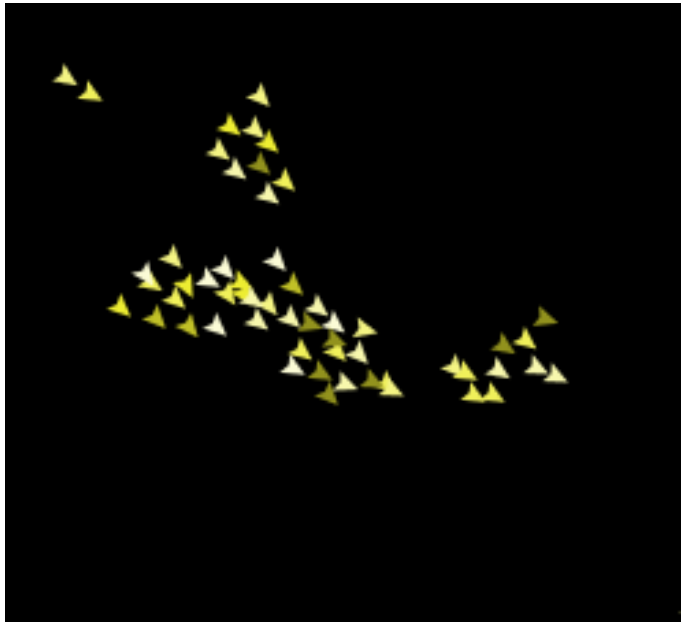
NetLogo: environement

NetLogo est un monde 2D constitué de

- Patches: constitue des "zones", des portions de l'environnement
- Turtles: (tortues) créatures qui peuvent se déplacer et agir dans cet environnement

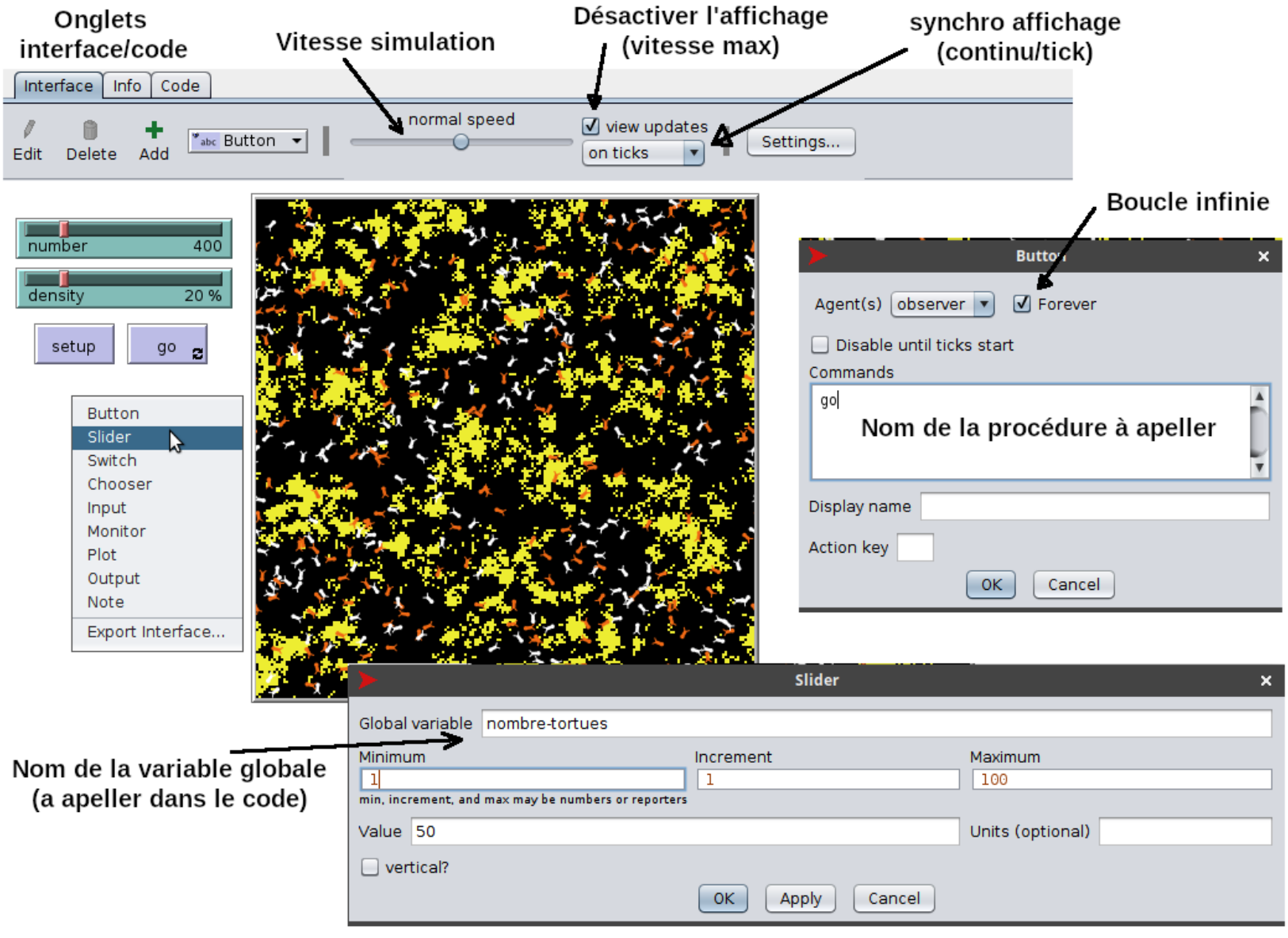


Tri termites



Vol en formation

NetLogo: interface



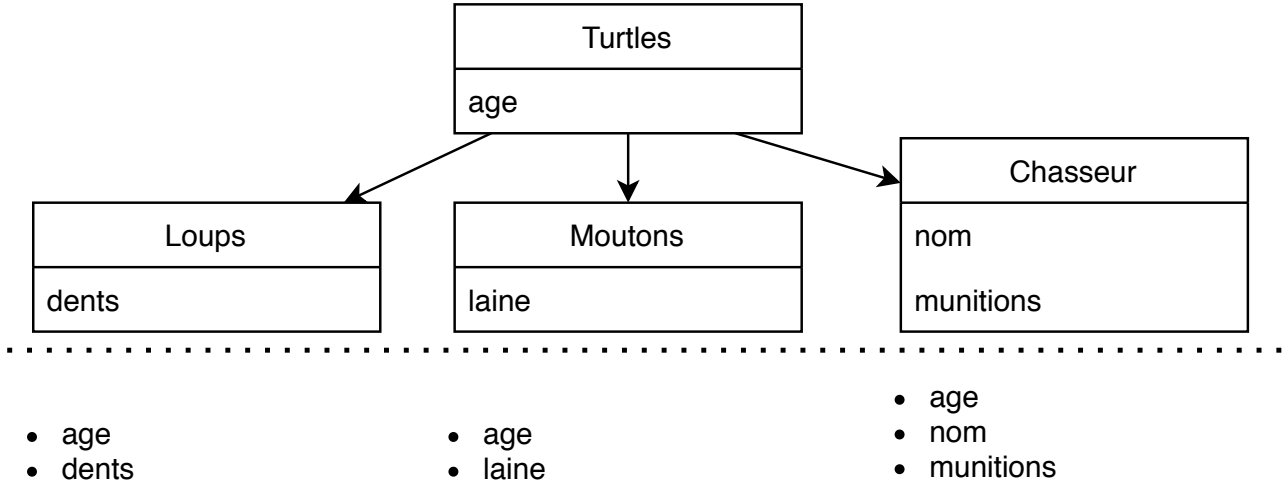
NetLogo: entités

Les patches

Se sont des parcelles de terrain. Elles ne bougent pas mais peuvent avoir un comportement et des attributs propres (ex: quantité d'herbe, type de terrain ...).

Les tortues

Capables de bouger, ce sont les agents d'un SMA.
Les tortues peuvent etre spécialisés en sous types apelés Breeds (espèces), qui héritent des attributs des tortures et peuvent ajouter les leurs. Un seul niveau d'héritage.



NetLogo: Structure minimale

```
globals [variableglobale]
turtles-own [attribut1 attribut2]
patches-own [attributpatch]

breed [wolves wolf]
wolves-own [attributloup]

to setup
  clear-all
  create-turtles 3 [
   set shape "bug"
   set attribut 0
  ]
  reset-ticks
end

to go
  ask turtles [ fd 1 ]
end
```

Déclaration des variables globales et attributs des tortues et patches

Déclaration d'une espèces et de ses attributs

Procédure pour initialiser (et réinitialiser) la simulation. Création de 3 tortues

Procédure d'exécution d'un pas de la simulation. Je demande a toutes les tortues de faire 1 pas.

NetLogo: procédures

Procédure sans retour, avec 1 paramètre

```
to draw-carre[taille]
   pen-down
   repeat 4 [fd taille rt 90]
end
```

Procédure avec retour et 1 paramètre

```
to-report absolute-value [ number ]
   ifelse number >= 0
      [ report number ]
      [ report 0 - number ]
end
```

NetLogo: structures contrôle

If

if *condition* [*instructions*]

```
1  if number >= 0 [ report number ]
```

If .. Else

ifelse *condition* [*instructions-then*][*instructions-else*]

```
1  ifelse number >= 0 [ report number ][ report 0 - number ]
```

Repeat

Pour répéter une instruction

repeat *nombre* [*instructions*]

```
1  repeat 4 [ fd 1 rt 90]
```

NetLogo: variables et attributs

Affectation

set *variable valeur*

```
set angle 90
set nom "carre"
```

Décalaration globale

```
globals [varGlobale1 varGlobale2 varGlobale3]
```

Décalaration attributs

```
turtles-own[att1 att2]
patches-own[att1 att2]
```

Décalaration locale

```
let varlocale
let varlocalebis 2
```

NetLogo: les tortues

breed [*pluriel singulier*] : sous-espèces

```
1 breed[ wolves wolf ]
```

create-turtles: créer et initialiser n tortues

```
1 create-turtles n [
2    set color green
3    setxy random-xcor random-ycor]
4 create-wolves n [
5    set dents "longues"]
```

ask : demander à un ensemble d'entités de faire quelque chose

```
1 ask turtles [ fd 1 rt 45]
2 ask patches [ set pcolor green]
```

NetLogo: déplacement

- fd *distance* : avancer (ForwarD)
- rt *angle* : tourner à droite (Right Turn)
- lt *angle* : tourner à gauche (Left Turn)

exemples pour tracer des figures

```
1  to carre [taille]
2     pendown    ;; commande pour commencer a tracer
3     repeat 4 [fd taille rt 90]
4  end
5  to spirale [taillePas nbPas anglePas]
6     pendown
7     repeat nbPas [
8        fd taillePas rt 360 / anglePas
9        set taillePas taillePas + 1 ]
10 end
```

NetLogo: orientation

set heading towards *entité*

- towards *entité* : calcule l'angle entre l'agent et la cible *entité*
- set heading : change l'attribut orientation de l'agent

un raccourcit : face *entité*

faire face à l'entité

exemples

```
1  set heading towards patch 0 0 ;;s'orienter vers le patch
       qui se situe au centre du terrain
2  face patch 0 0 ;;equivalent
3
4  ;;recuperer une des tortues a proximite
5  let v one-of other turtles in-radius 3
6  ;;si il y a une tortue, lui faire face
7  if (v != nobody)
8     [face v]
```

NetLogo: AgentSets

Un sous ensemble d'entités (patches ou tortues)

```
1 turtles with [color = red ]
2 patches with [pxcor > 0]
3 turtles in-radius 3
4 turtles in-radius 3 with [color = red ]
5 turtles with [color = red ] in-radius 3 ;; quelle est la
      difference a votre avis ?
```

aux éléments duquel on peut demander quelque chose:

```
1 ask turtles with [color = red] [fd 10]
```

NetLogo: AgentSets

selectioner un élément d'un agentSet

```
one-of turtles in-radius 3
```

au hasard

```
one-of other turtles in-radius 3
```

au hasard, mais en excluant moi même

```
max-one-of turtles [age]
```

avec l'attribut le plus grand

test de présence

```
any? turtles in-radius 3
```

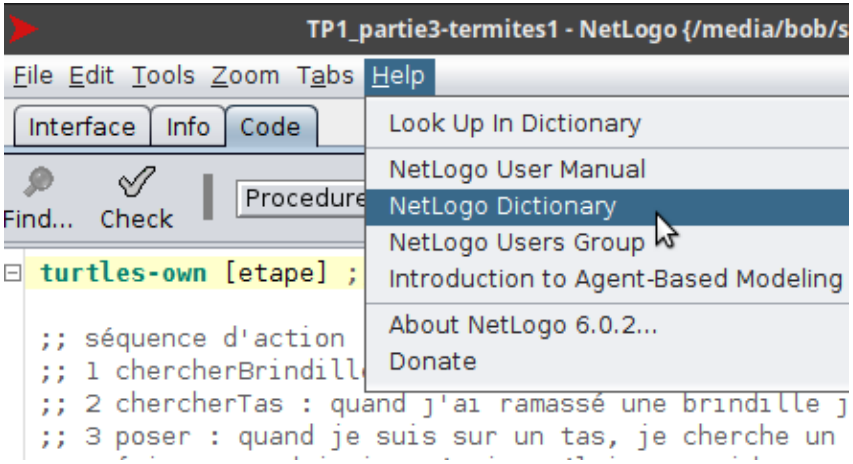
si un agentSet n'est pas vide

```
let v one-of turtles in-radius 3
if v != nobody
[;; il existe une tortue v ]
```

si un élément selectioné n'est pas NULL

Bien programmer en suivant le protocole RTFM

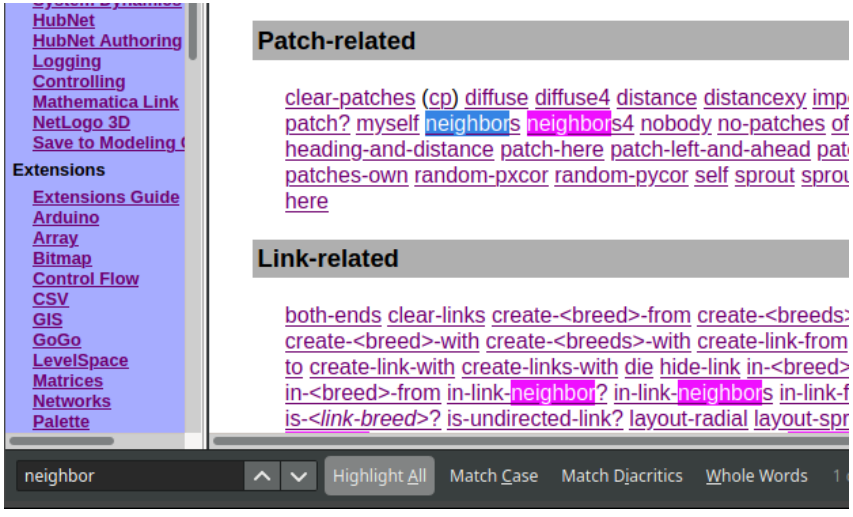
J'aimerais savoir comment récupérer les patches autour de celui ou je suis.

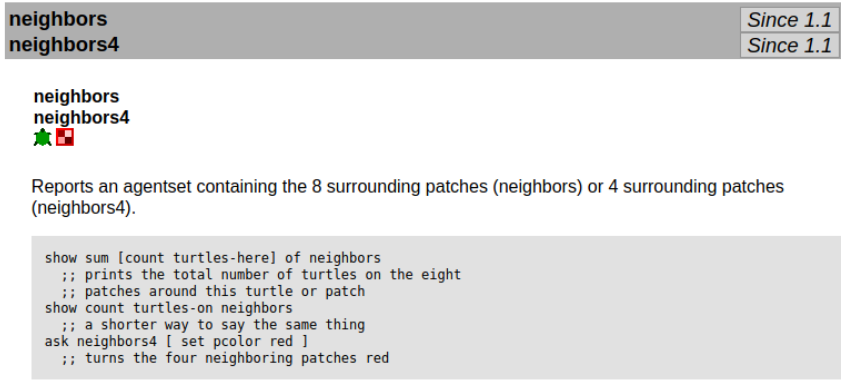


Trop d'information ...

Bien programmer en suivant le protocole RTFM

1. Les patches autour en un mot ... voisins ?
2. en anglais : neighbors
3. ctrl + f rechercher neighbors
4. je trouve un lien neighbors dans la catégorie Patch-related





NetLogo: exemple termites

```
1  turtles-own [etape] ;; un attribut pour representer
       l'etape du comportement de la termite
2
3  to setup
4    clear-all
5    set-default-shape turtles "bug"
6    ;; randomly distribute wood chips
7    ask patches
8    [ if random-float 100 < density
9      [ set pcolor yellow ] ]
10   ;; randomly distribute termites
11   create-turtles number [
12     set etape "chercher"
13     set color white
14     setxy random-xcor random-ycor
15     set size 5 ;; easier to see
16   ]
17   reset-ticks
18 end
```

NetLogo: exemple termites

```
1  ;; pour une jolie simulation, faites en sorte que vos
       agents ne "jouent" qu'une seule fois par tick (evitez
       les appels recursifs!)
2  to go
3    ask turtles
4    [
5    ifelse etape = "chercherBrindille"
6    [search-for-chip]
7    [ifelse etape = "chercherTas"
8    [find-new-pile]
9    [ifelse etape = "poser"
10   [put-down-chip]
11   [ifelse etape = "fuir"
12   [get-away]
13   [set etape "chercherBrindille"]]]]
14   ]
15   tick
16 end
```

NetLogo: exemple termites

```
1  to search-for-chip
2    ifelse pcolor = yellow [
3      set pcolor black
4      set color orange
5      fd 20
6      set etape "chercherTas"]
7    [ wiggle ]
8  end
9
10 to find-new-pile
11   ifelse pcolor != yellow
12   [ wiggle ]
13   [ set etape "poser" ]
14 end
```

NetLogo: exemple termites

```
1  to put-down-chip
2    ifelse pcolor = black
3    [ set pcolor yellow
4      set color white
5      set etape "fuir" ]
6    [ rt random 360
7      fd 1 ]
8  end
9
10 to get-away
11   rt random 360
12   fd 20
13   if pcolor = black
14   [ set etape "chercherBrindille" ]
15 end
16
17 to wiggle
18   fd 1  rt random 50 lt random 50
19 end
```