SYSTÈMES MULTI-AGENTS ET SIMULATION INTERACTIVE

Systèmes complexes &
Modélisation et Simulation
à base d'agents

Jérémy Rivière
jeremy.riviere@univ-brest.fr

Département d'Informatique
Equipe IHSEV, Lab-STICC

Plan

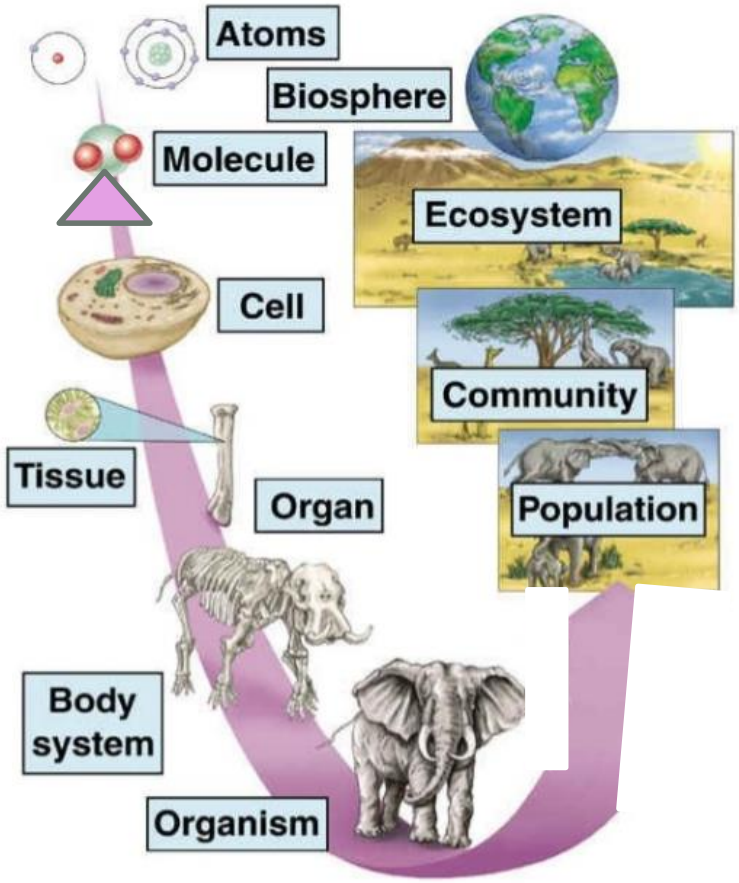
1. Introduction aux systèmes complexes
2. Modélisation et simulation
3. Les SMA pour la simulation et la modélisation de systèmes complexes
4. Visualiser et interagir avec des simulations à base d’agents

Les systèmes complexes
Complexité ?

Un exemple :
la complexité biologique

- Comment l'appréhender ?

Réductionnisme depuis 300 ans



Les systèmes complexes
Complexité ?

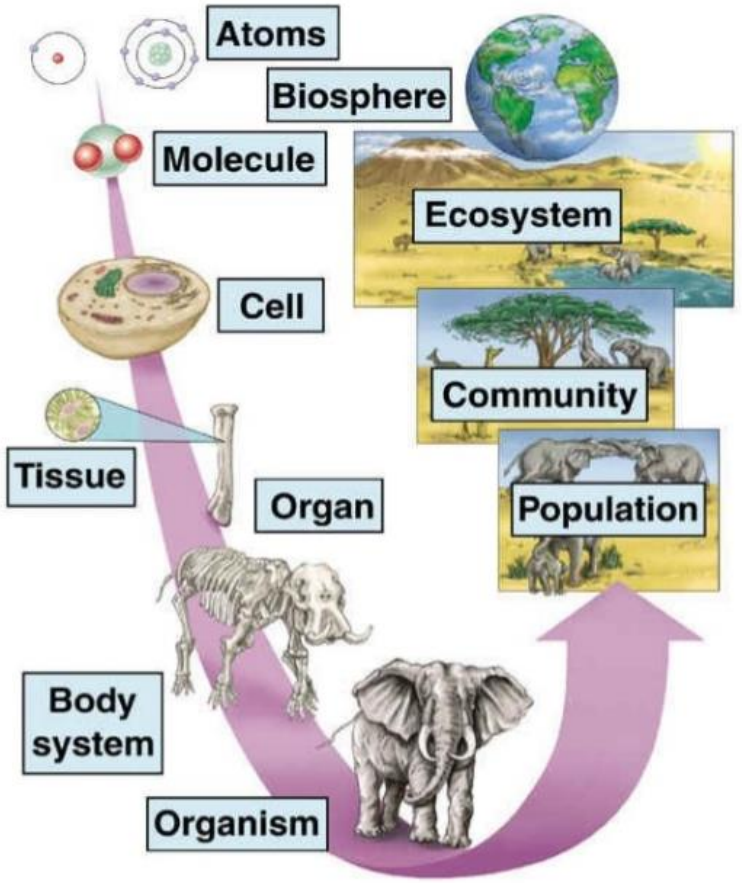
Un exemple :
la complexité biologique

- Comment l'appréhender :

Réductionnisme

- Ok, mais comment on « remonte ? »

- C'est **complexe** : interdépendances, auto-organisation, feedbacks, propriétés *émergentes* …



Les systèmes complexes
Complexité ?

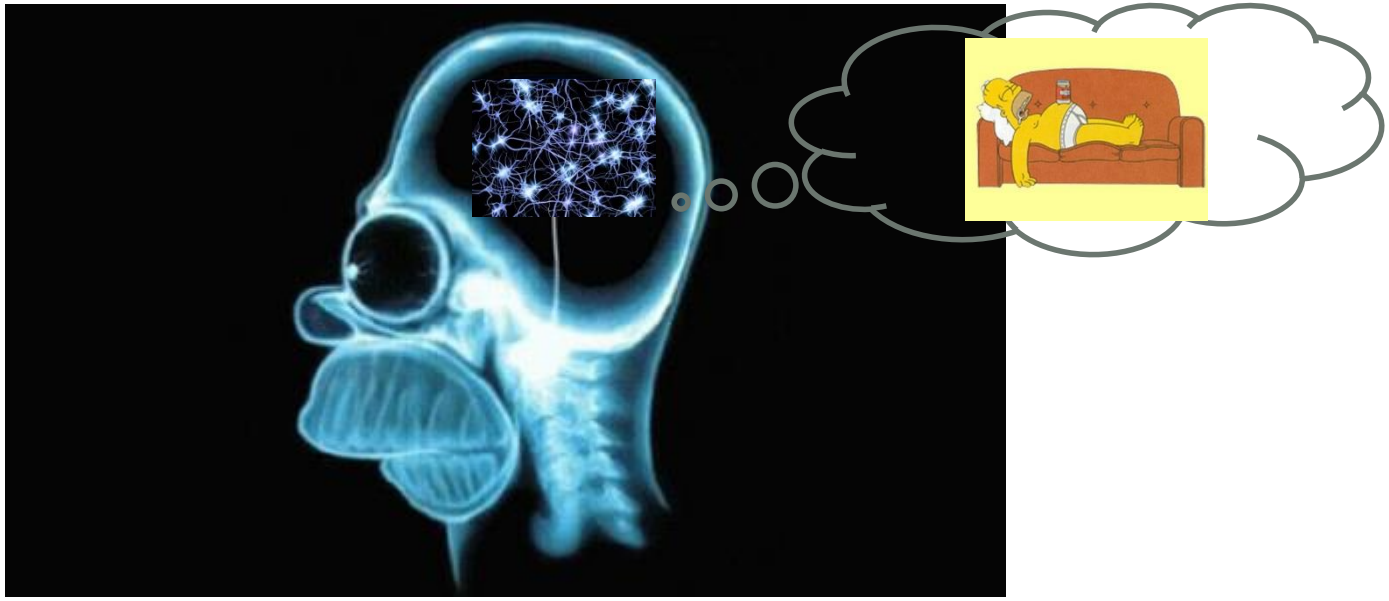
Un autre exemple : le **cerveau**

- environs 10^{11} neurones
- interconnectés, communiquant par impulsions électriques et chimiques

Les systèmes complexes
Complexité ?

Un autre exemple : le **cerveau**

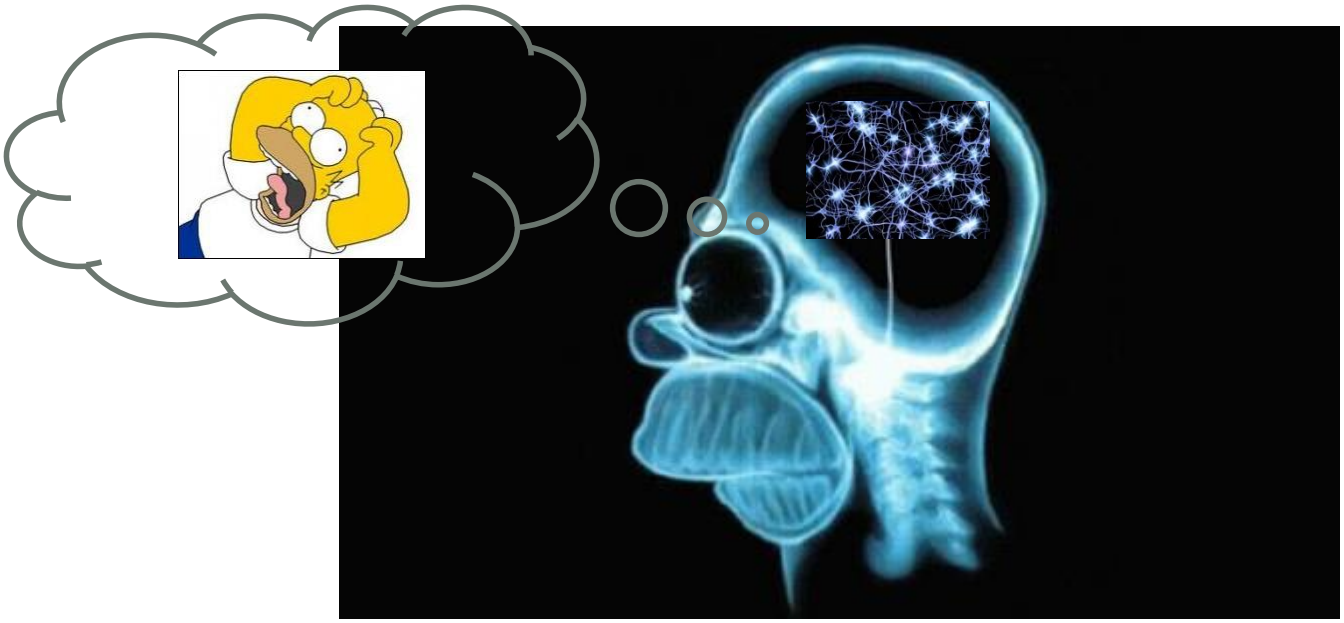
- environs 10^{11} neurones
- interconnectés, communiquant par impulsions électriques et chimiques



Les systèmes complexes
Complexité ?

Un autre exemple : le **cerveau**

- environs 10^{11} neurones
- interconnectés, communiquant par impulsions électriques et chimiques



Complexité : un enjeu scientifique

LA question à laquelle essaie de répondre la science des systèmes complexes est :

Comment de grands systèmes

- composés de parties "simples",
- avec une communication limitée entre ces parties,
- et sans "leader" ou contrôle central

peuvent-ils montrer des **comportements adaptatifs, organisés, émergents,** bref, **complexes**?

« *The whole is greater than the sum of its parts* » - Aristote

Une question concrète

Une question concrète

Pourquoi ?

Quelques hypothèses :

- Prédateurs
  - un seul (grand et menaçant) organisme
  - plus difficile de cibler un seul individu
- Plus efficace dans la prédation (chasse coopérative)
- Améliore l'aérodynamisme des individus (cyclisme !)

Beaucoup d'hypothèses, non-exclusives

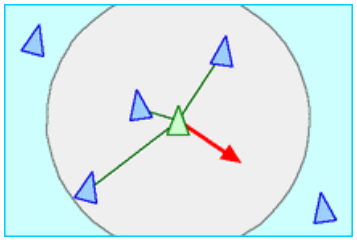
Une question concrète

Comment ?? (sans leader, avec seulement des informations locales)

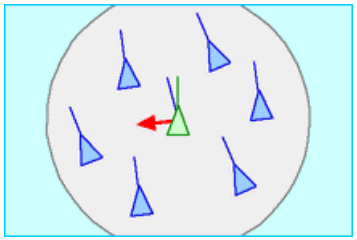
Modèle "Boids" *[Craig Reynolds, 1987]* :

Seulement 3 règles pour **le comportement de chaque oiseau**, dans l'ordre d'importance :

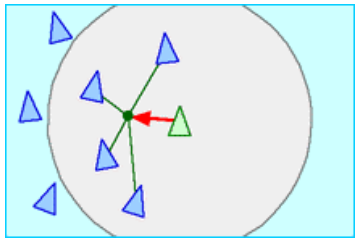
1. **Eviter les collisions** avec les oiseaux les plus proches



2. Essayer **d'égaler la vitesse et la direction** des oiseaux les plus proches



3. Essayer de **rester proche** des oiseaux les plus proches



Une question concrète

Comment ?? (sans leader, avec seulement des informations locales)

Modèle "Boids" *[Craig Reynolds, 1987]* :

Seulement 3 règles pour le comportement de chaque oiseau, dans l'ordre d'importance :

1. Eviter les collisions avec les oiseaux les plus proches
2. Essayer d'égaler la vitesse et la direction des oiseaux les plus proches
3. Essayer de rester proche des oiseaux les plus proches

Flocking.nlogo

Et ensuite ? **Simulation informatique**, par exemple avec NetLogo *(http://ccl.northwestern.edu/netlogo/)*

⟹ **Expérimenter le modèle, tester d'autres hypothèses**

Définition

Un système complexe

- est constitué de beaucoup de **différentes** parties **interconnectées** et **interagissant**
- n'a pas de **contrôle central**
- est **multi-échelle**
- est **ouvert et dynamique**

Les systèmes complexes sont composés d'un grand **nombre** et d'une certaine **diversité** d'**individus/entités**, situés dans un **environnement**, **interconnectés, autonomes** et **interagissant** entre eux et avec l'environnement.

Propriétés importantes

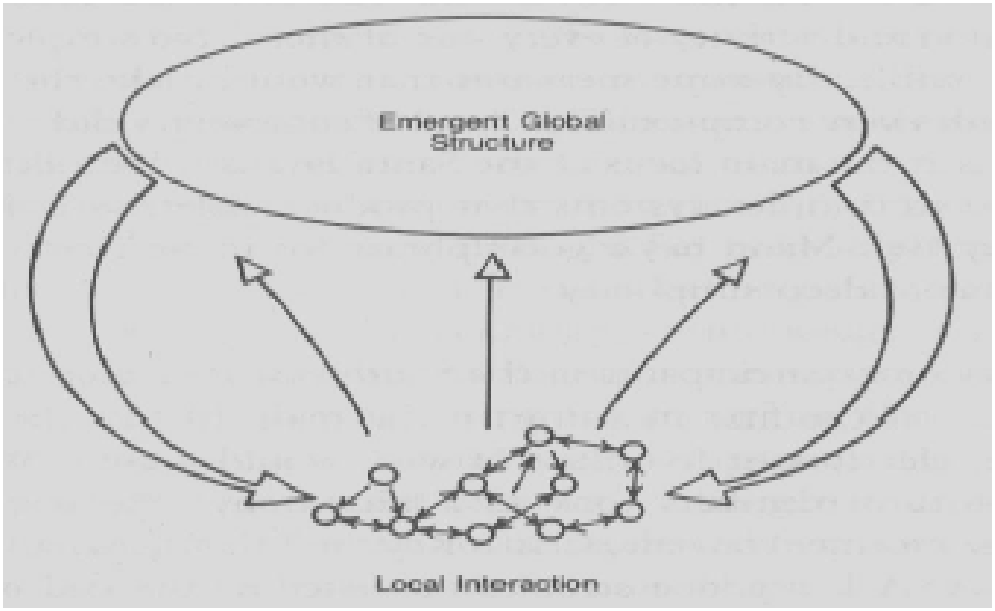
Quelques concepts fondamentaux et propriétés :

- **Emergence**
  - L'extraction de nouvelles propriétés, structures, patterns, à partir du système complexe
  - Les phénomènes émergents proviennent des interactions non-linéaires, distribuées, entre les entités du système
  - Ces phénomènes sont observables à un niveau « macro », même s'ils sont générés par des interactions au niveau « micro »

Propriétés importantes

Quelques concepts fondamentaux et propriétés :

- **Emergence**
  - Le système (macro) a des propriétés que ses éléments (micro) n'ont pas
  - Les phénomènes émergents peuvent influencer (feedback) les éléments à leur tour : e.g. embouteillages, marchés financiers …



Chris Langton's view of emergence in complex systems
(from "Complexity", Roger Lewin, University of Chicago Press)

Propriétés importantes

Quelques concepts fondamentaux et propriétés :

- **Emergence**
- **Auto-organisation**
  - Changements dans l'ordre interne ou dans l'organisation du système sans guidage ou gestion d'une source extérieure
  - Nuées d'oiseaux, sélection naturelle, réseaux de neurones …

  - Par interactions locales
  - directes : élément <-> élément
  - indirectes : élément <-> environnement <-> élément (stigmergie chez les insectes sociaux)

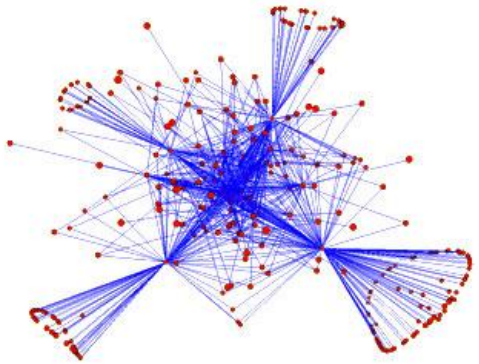
  - Différent de l'émergence !

Propriétés importantes

Quelques concepts fondamentaux et propriétés :

- **Emergence**
- **Auto-organisation**
- **Auto-adaptation**
- **Décentralisation**
- **Non-linéarité** (part de hasard, dépendance au chemin)
- « **Effet papillon** » : petites actions, grandes conséquences

Loutres de mer et forets de kelp



https://www.youtube.com/watch?v=ysa5OBhXz-Q

Propriétés importantes

Exemple : Fourmilières et « autoroutes » de fourmis

- Emergence de structures et d'un comportement global
- Décentralisation
- Auto-organisation des fourmis : rôles dans la fourmilière, recherche de nourriture, construction du nid …
- Auto-adaptation du système en cas de perturbation (attaque, famine …)
- Non-linéarité (*e.g.* dans la recherche aléatoire de nourriture)

-> chemin le plus court !!

Systèmes complexes et interdisciplinarité

On trouve des systèmes complexes dans toutes les sciences

- Introduisent des **principes universels** qui peuvent être utilisés pour décrire et résoudre des problèmes de la **physique** des particules à **l'économie**, la **biologie**, la **chimie**, les **sciences sociales** …
- Exemples ?
- Ils font donc le **lien** entre chimie, physique, biologie, anthropologie, sociologie, informatique ...

⇒ Transfert des idées et des résultats entre ces différents champs de recherche

cnrs

Institut des Systèmes Complexes Paris Ile de France

COMPLEX SYSTEMS SOCIETY

Comprendre les systèmes complexes
Le rôle des modèles

Jacques Tisseau

« Describing, explaining, predicting and simulating the behavior of natural or artificial complex systems is, for scientists, one of the major challenges of the 21st century »

- La compréhension et / ou la mise en place d'un système complexe dépend de la **modélisation**
- Etude de ces modèles à travers **la simulation**
- Simulation ***in virtuo*** : **visualiser** le comportement du modèle et **interagir** avec lui

Plan

1. Introduction aux systèmes complexes

2. Modélisation et simulation
   1. Qu'est-ce qu'un modèle ? Pourquoi modéliser et simuler ?
   2. Principes généraux de modélisation
   3. Ingénierie de la modélisation : processus et stratégies particulières
   4. Quel paradigme pour les systèmes complexes ?

3. Les SMA pour la simulation et la modélisation de systèmes complexes

4. Visualiser et interagir avec des simulations à base d’agents

Qu'est qu'un modèle ?

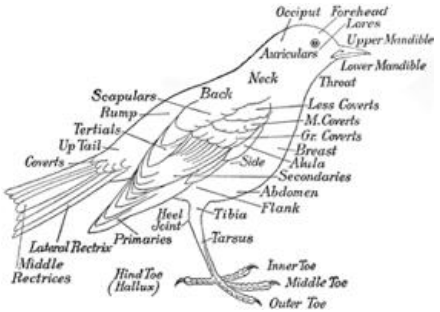
- Un modèle est une représentation simplifiée de la réalité, un outil de réflexion
- Un (ou plusieurs) aspect(s) du système réel
- Exemples : maquette, dessin, robot, équation …

Oiseau réel

MODELISATION

Maquette volante

Robot digérant



Description schématique

"Modélisation" et "simulation"

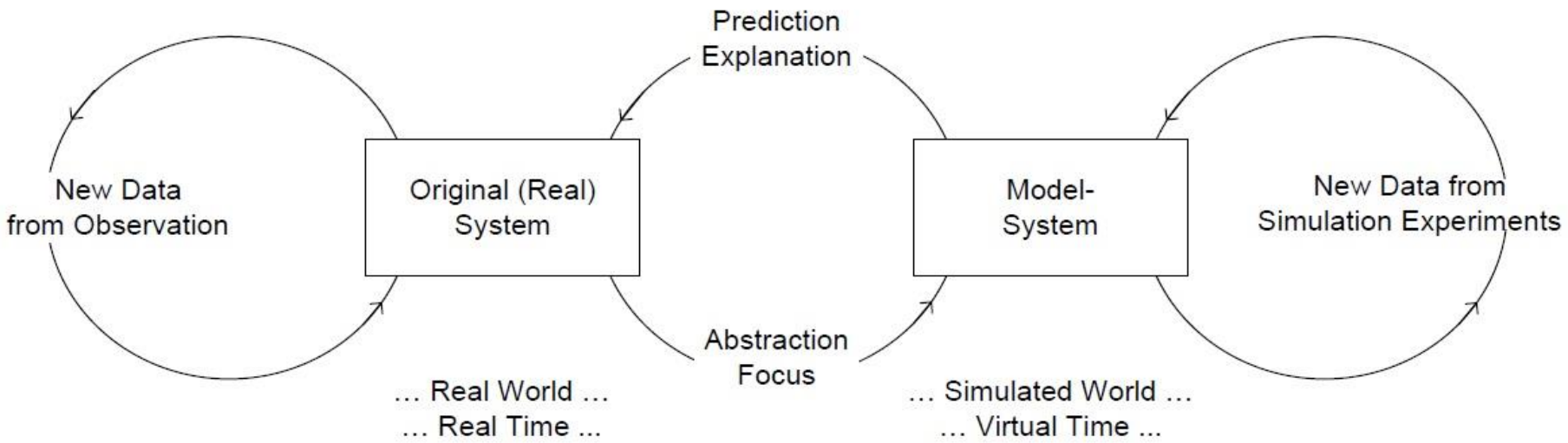
- Modéliser, c'est

- adopter un point de vue partial sur une représentation de la réalité
- un processus abstrait
  - Aspects computationnels (*e.g.* gestion du temps)
  - Aspects technologiques (*e.g.* plateforme, langage)

- Simuler, c'est **expérimenter le modèle** en le "plongeant" dans le temps, dans le but de

- comprendre le système réel
- expérimenter, obtenir un résultat
- prédire l'évolution du système réel
- calibrer et valider le modèle dans une approche expérimentale

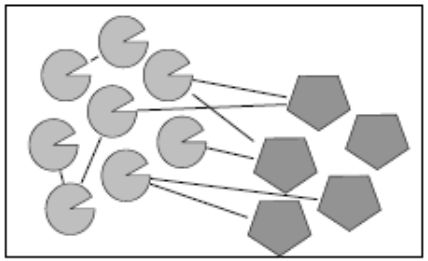
"Modélisation" et "simulation"

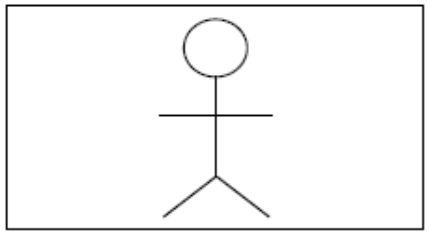


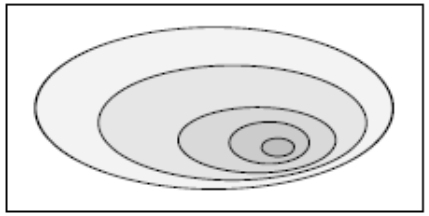
Tirée de [F. Klügl 2016]

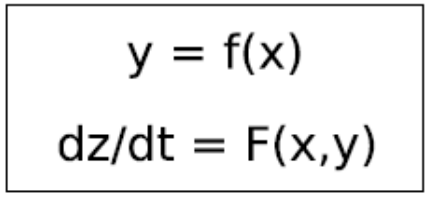
Paradigmes de modélisation

[Bonneau et al. 2007]









Modèles	Solutions
Modèles émergents à base d'agents (ABM, IBM) *"bottom-up"*	Emergent du comportement et des interactions des agents
Modèles participatifs joués par un expert	Données par l'expert
Modèles étendus Configurations connues, basées sur des données	Toutes les configurations construite à partir des données
Modèles intentionnels Modèles populationnels, à base d'équations (PDE) *"top-down"*	Toutes les configurations qui satisfont les équations

Pourquoi …

… Modéliser et simuler <u>en général</u> ?

- La partie du système réel à étudier n'est pas accessible
- Expérimenter sur le système réel est interdit
- L'échelle de temps du système réel est trop petite, ou trop grande, pour permettre son observation
- Contrôle de l'ensemble des variables et de l'environnement de la simulation
- Concrétiser une théorie scientifique par expérimentation virtuelle
- Prévoir par simulation du modèle

Météorologie -> modèle du climat

Pourquoi …

… Modéliser et simuler les systèmes complexes ?

- Le seul moyen de comprendre, décrire, expliquer le **comment** des propriétés émergentes, auto-*

> « *Ce que je n'arrive pas à créer, je ne le comprends pas* »
> Richard Feynman

- Mettre en place un système complexe artificiel …
- … En intégrant l'humain en interaction

Construction d'un modèle : 5 Principes

[Pidd, 1996]

1. **Model Simple, Think Complicated**
   Les modèles doivent être des outils de réflexion, des représentations simplifiées du système

2. **Be Parsimonious : Start Small and Add**
   A quel point le modèle d'un système peut-il être simple ? KISS

3. **Divide and Conquer : Avoid Mega-Models**
   Plutôt une composition de modèles (dans des modules) plutôt qu'un seul modèle très grand, difficile à valider, interpréter, calibrer et expliquer

4. **Do not Fall in Love with Data**
   Les données ne doivent servir qu'à calibrer le modèle

5. **Model Building May Feel Like Modeling Through**
   Pas un processus linéaire, structuré ;
   90% modélisation, définition du problème … 10% implémentation

Stratégies de modélisation
KISS

"Keep It Simple, Stupid"

- Garder le modèle **le plus simple possible** pour générer le comportement voulu
- Vaut pour la création du modéle comme pour le choix du paradigme
- Résultat (même "juste" !) parfois éloigné du système réel

1. Identifier et décrire l'ensemble S des propriétés observables (patterns) du système réel
2. Définir un modèle M_0 apparemment trop simple pour reproduire toutes les propriétés du système
3. Par calibration, déterminer l'ensemble S_M des propriétés reproduites par M_0
4. M <- M_0
5. Répéter jusqu'à ce que S_M = S
   a) modifier M pour produire plus d'éléments dans S que dans l'itération précédente
   b) par calibration, déterminer l'ensemble S_M des propriétés reproduites par M

Stratégies de modélisation
KIDS

"Keep It Descriptive, Stupid"

- Garder le modèle **le plus complet possible** pour générer le comportement voulu
- Beaucoup de paramètres dans les systèmes complexes, difficiles à identifier

1. Répéter jusqu'à ce qu'un modèle valide et crédible M_s soit construit
   a) Définir un modèle M contenant tous les aspects apparemment importants du système réel
   b) Identifier toutes les hypothèses et affirmations et expliciter tous les paramètres de M
   c) Faire une analyse de sensibilité des paramètres pour tous les paramètres de M et éliminer tous les blocs de comportement contrôlés par un paramètre qui n'a pas d'effet sur le comportement global.
   d) M_s <- M
   e) Vérifier la crédibilité et la validité de M_s

Stratégies de modélisation
TAPAS

"Take A Previous model, Add Something"

- Pragmatique, réutilisation de modèles existants
- KIDS à partir d'un modèle existant
- Problème de la description du modèle existant (détails, paramètres …) et de la disponibilité des données pour validation

1. Sélectionner un modèle existant **M** *approprié* au problème
2. Si **M** n'est pas implémenté, l'implémenter et le valider à partir des données publiées sur **M** (si disponibles)
3. Ajouter de nouveaux aspects pour produire M_{add}
4. Tester et valider M_{add}

Retourner à l'étape 3, ou 1, si non satisfaisant

Modéliser et Simuler
Processus global

Etape 1 : Modéliser

1. Point de départ : **une question scientifique**
   - problème et positionnement
   - un système ciblé
   - des données (pour la **calibration** et la **validation** du modèle)
   - une hypothèse scientifique

2. Une cadre théorique

3. Des phénomènes observés, des patterns, qu'on devra reproduire

4. Vérification du modèle (experts)

-> Un modèle conceptuel, suivant un paradigme, du système ciblé

Modéliser et Simuler
Processus global

Etape 2 : Simuler

1. Des questions de simulation
   - Quelles sont les réponses attendues ?
   - Quelles données utiliser ?
2. Une implémentation du modèle
3. Un protocole expérimental
4. Analyse des biais et de la sensibilité des paramètres
5. Calibration

Wolf Sheep Predation.nlogo

Modéliser et Simuler
Processus global

Etape 3 : Analyser les résultats

1. **Représentation** des résultats : preuves de l'observation des phénomènes, des patterns simulés

2. **Validation** du modèle

3. **Exploitation** des résultats : réfutation ou validation de l'hypothèse

4. **Reformulation** de la question scientifique initiale

5. **Enrichir le cadre théorique** avec de nouvelles hypothèses

6. **Publier** un article scientifique !

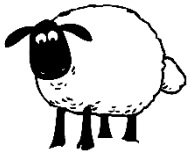
Description du modèle ??

Quel paradigme de modélisation pour les systèmes complexes ?

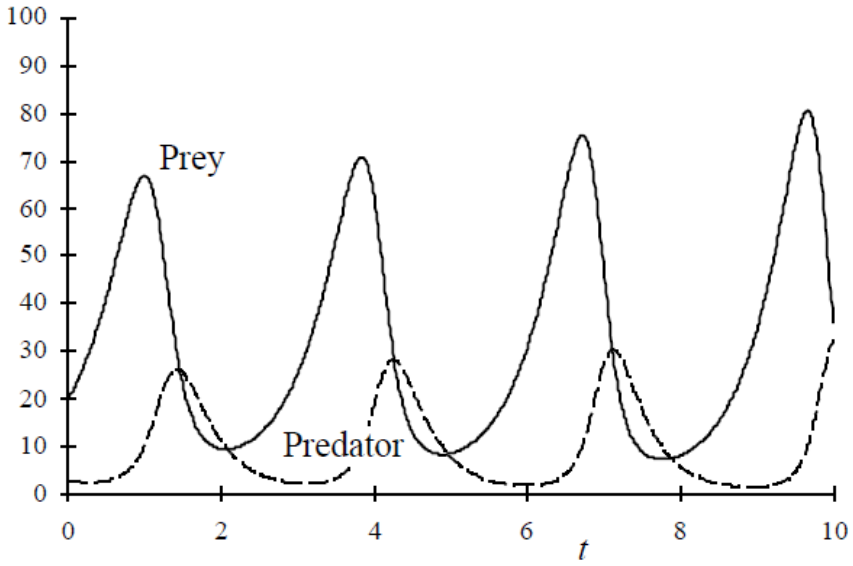
Un exemple
Ecosystème : les systèmes proie – prédateur

- Prédateurs :
  - se nourrissent de proies
  - se reproduisent
  - meurent de faim

- Proies :
  - se nourrissent d'herbe
  - se reproduisent
  - meurent de vieillesse



Phénomène **émergent** observé :



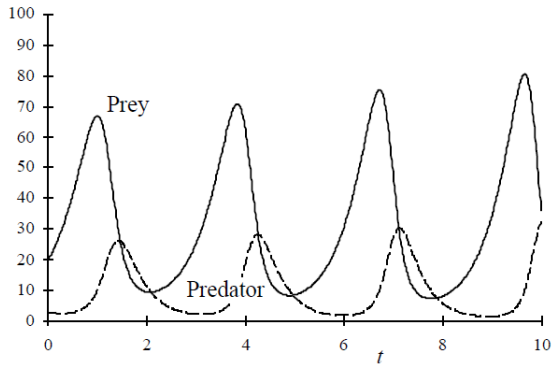
Quel paradigme de modélisation pour les systèmes complexes ?

Ecosystème : les systèmes proie – prédateur

- Approche « ***top-down*** » : équations différentielles

$$\frac{da}{dt} = \lambda_a a - \gamma ab$$

$$\frac{db}{dt} = -\mu b + \lambda_b \gamma ab$$



⇨ ne prend pas en compte :

- L'espace/l'environnement (l'emplacement des proies, de l'herbe, des clôtures, les périodes de sécheresse …)
- La diversité des individus

⇨ ne permet pas d'expliquer :

- Les différents mécanismes qui forment le phénomène émergent

Quel paradigme de modélisation pour les systèmes complexes ?

Un autre exemple

Chimie : la formation de l'eau à partir des atomes d'hydrogène et d'oxygène

- Les propriétés de l'eau (densité, l'élasticité, changement d'états, le mouvement) n'apparaissent pas dans les atomes, et ne peuvent être déduites de leur seule description
- Approche « *top-down* » : équations décrivant ces propriétés
- Approche « ***bottom-up*** » ?

Quel paradigme de modélisation pour les systèmes complexes ?

Encore un exemple

Sociologie : la ségrégation

- A Los Angeles, malgré **une faible tendance** des habitants des différents quartiers pour la ségrégation, on voit **apparaitre** (*émerger*) **des quartiers bien définis par ethnie**. Comment l'expliquer ?

Modèle de Thomas Schelling, 1971 :

- Il suffit d'une préférence de 30% concernant **son** voisinage pour faire apparaitre des quartiers distincts.

- Approche « bottom-up »

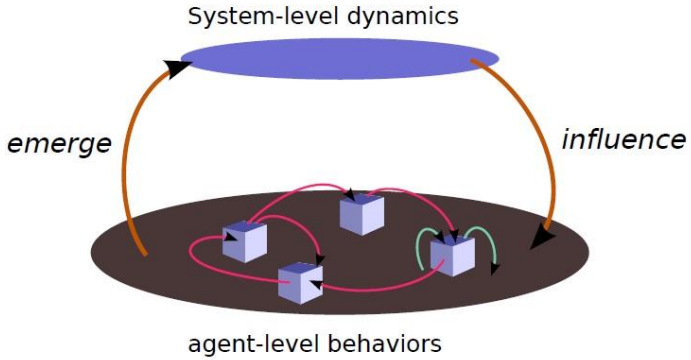
Segregation.nlogo

Modèles à base d'agents
A utiliser quand …

- Les agents sont une **métaphore naturelle**
  - Facile d'identifier les agents comme les entités individuelles d'un système
- Les comportements individuels sont plus **simples** que le comportement du système global
- **Interactions nombreuses** et fortes entre les entités
- Le comportement des entités **dépend du contexte**
  - Propriétés locales de l'environnement, voisinage, état interne …
- On se concentre sur les **propriétés émergentes** du système
  - Liens entre les processus à différentes échelles (micro-macro)

Sur quoi met l'accent l'approche à base d'agents ?

- Variabilité interindividuelle, **hétérogénéité** des entités
- **Interactions locales**
- **Cycle de vie** des individus
- **Plasticité** des comportement, l'évolution des entités
- Influence du comportement des individus sur le système (**émergence**) et du système sur les individus (***immergence***) – lien macro-micro

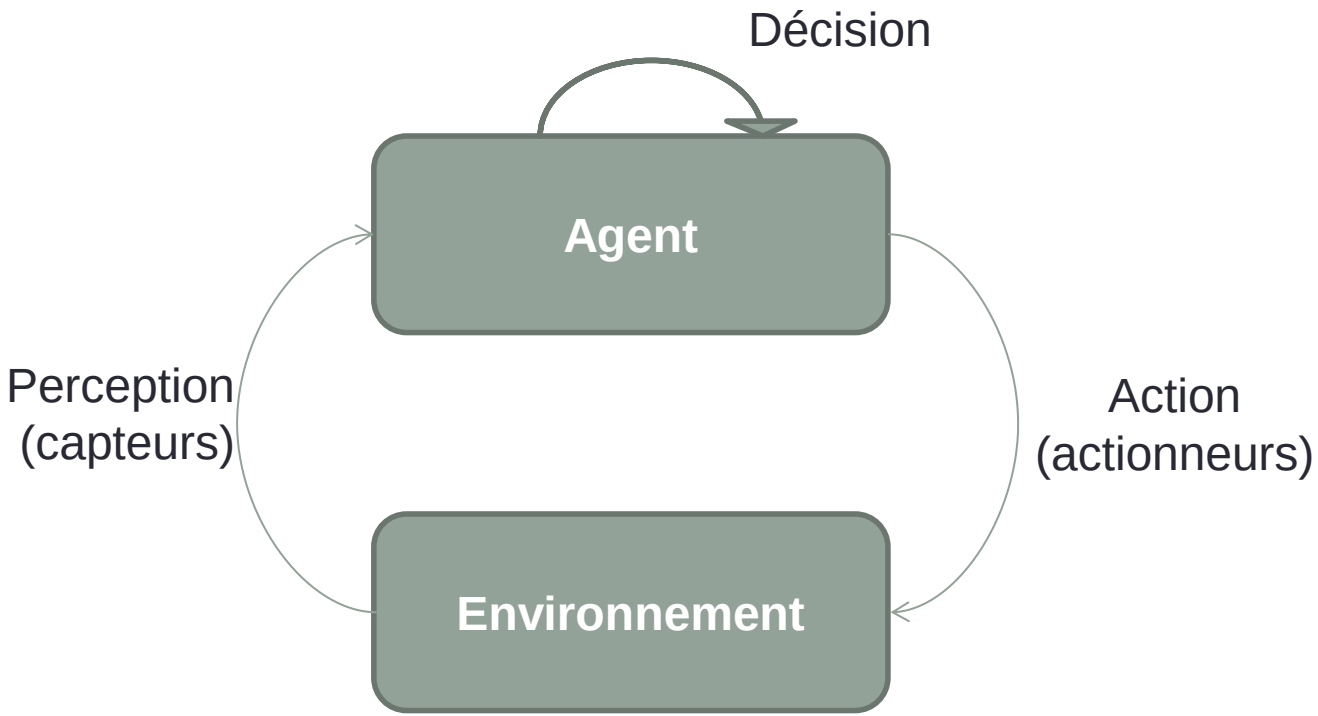


Plan

1. Introduction aux systèmes complexes
2. Modélisation et simulation
3. Les SMA pour la simulation et la modélisation de systèmes complexes
    1. Rappels et définition
    2. Principes de modélisation SMA : étude de cas
    3. Challenges et bonnes propriétés d'un modèle
    4. Le temps dans les simulations
    5. Implémentations des SMA
        1. Principales plateformes de simulation
        2. Implémentation Objet en Java
4. Visualiser et interagir avec des simulations à base d’agents

Définition d'un agent

Un agent est vu comme une entité **autonome**, qui peut **percevoir** et **agir** dans son **environnement**, et qui est capable **d'interagir** avec d'autres agents



Définition d'un agent
Capacités de perception et d'action

Perception et actions dans **l'environnement** physique, virtuel et/ou social

- partiellement connu par l'agent (inaccessible *vs.* accessible)
- non-déterministe (*vs.* déterministe)
- dynamique (*vs.* statique)
- continu (*vs.* discret)
- ouvert (*vs.* fermé)

[Russel and Norvig, 2003]

Système complexe ?

Définition d'un agent
Capacités **limités** de perception et d'action

Perception et actions dans **l'environnement** physique, virtuel et/ou social

Perceptions et actions locales

- entités pas omniscientes
- encore plus vrai dans un système complexe

Un répertoire limité d'actions

- agents avec des capacités différentes

Une quantité limitée de temps et de ressources pour prendre la décision

- agir dans un environnement dynamique

Définition d'un agent
Capacités d'interactions

Interaction comme actions et perceptions

- entre les agents (dont les humains)
- entre l'agent et l'environnement (influence et réactions)
- de et dans l'environnement

Différents types d'interactions

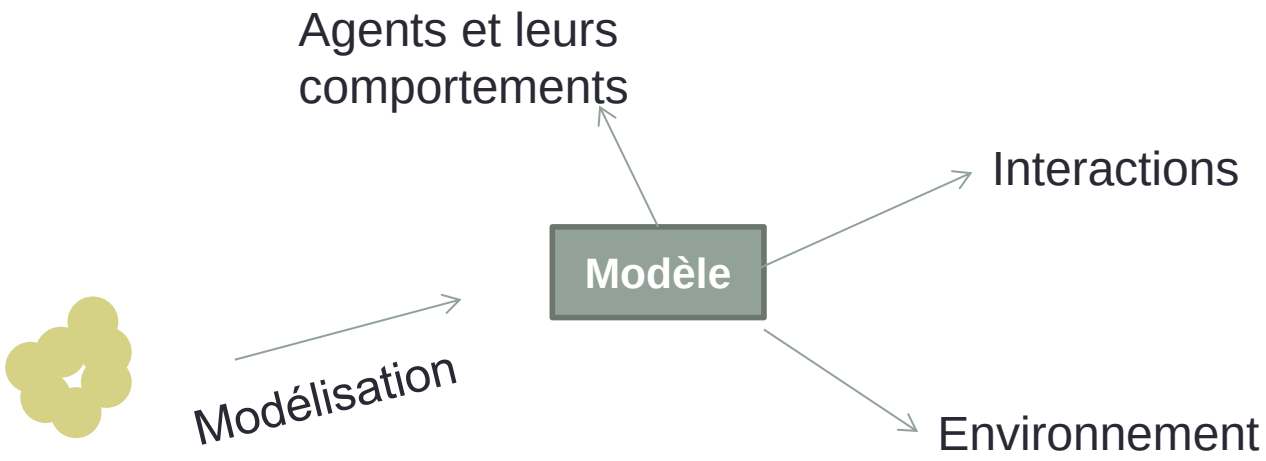
- Communication directe ou indirecte
- Situations de collaboration, coopération, compétition

Organisation des agents

- Induit des contraintes, des règles sur les interactions
- Exemple : simulation d'une colonie d'abeilles

SMA pour la modélisation

Modéliser un phénomène complexe suivant une approche multi-agents revient à identifier :



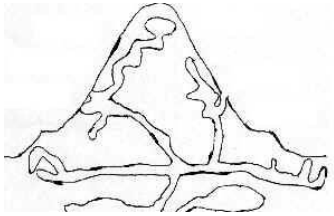
Exemple de simulation SMA
Les « autoroutes » de fourmis

- *Comment expliquer la formation « d'autoroutes » de fourmis du nid vers une source de nourriture ?*

- Données de la biologie
  - Comportements observés de fourmis
  - Capacités de perception et d'action
  - Communication très simple par phéromones
  - Evaporation des phéromones au cours du temps

Exemple de simulation SMA
Les « autoroutes » de fourmis

Environnement

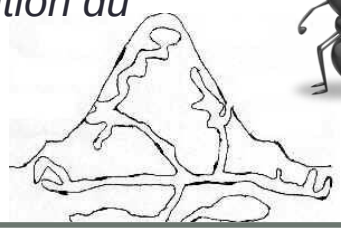


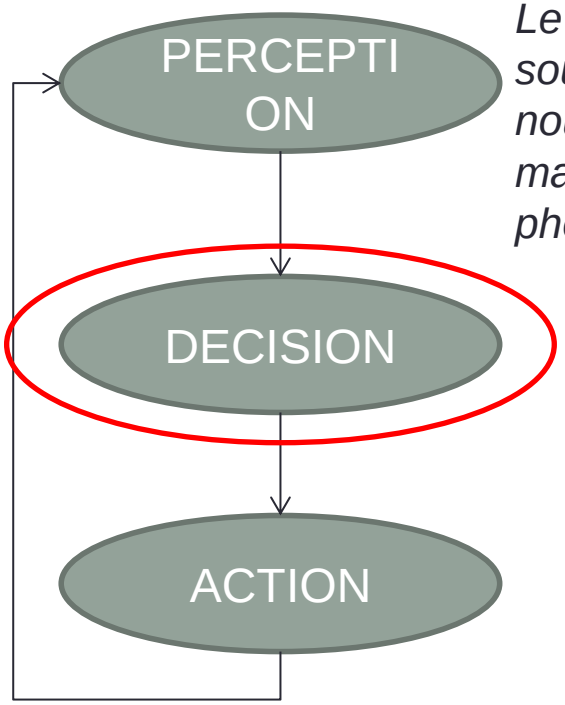
Exemple de simulation SMA
Les « autoroutes » de fourmis

Environnement

Chaque agent représente une fourmi

Attributs :
Chargée,
Position du nid,
…

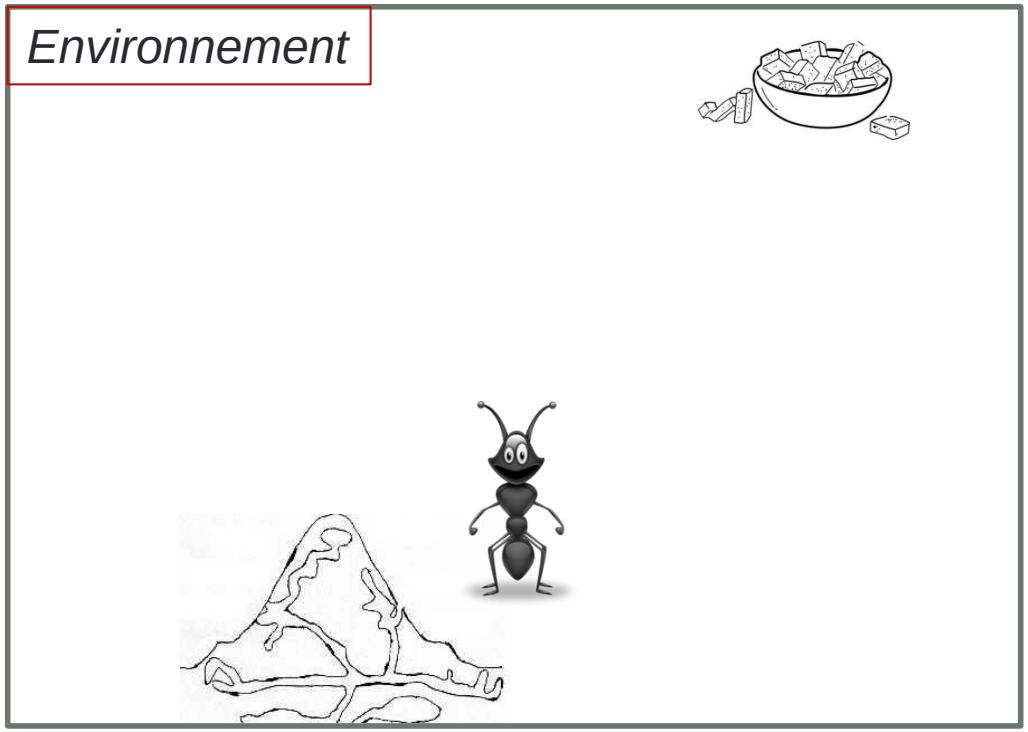




Le nid, les sources de nourriture, les marques de phéromones

Se déplacer, Ramasser / Déposer nourriture, Déposer une marque de phéromone

Exemple de simulation SMA
Les « autoroutes » de fourmis

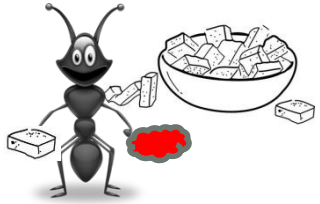


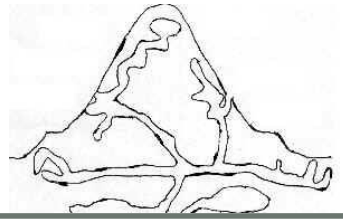
Règles de décision de l'agent :

- Explore aléatoirement

Exemple de simulation SMA
Les « autoroutes » de fourmis

Environnement

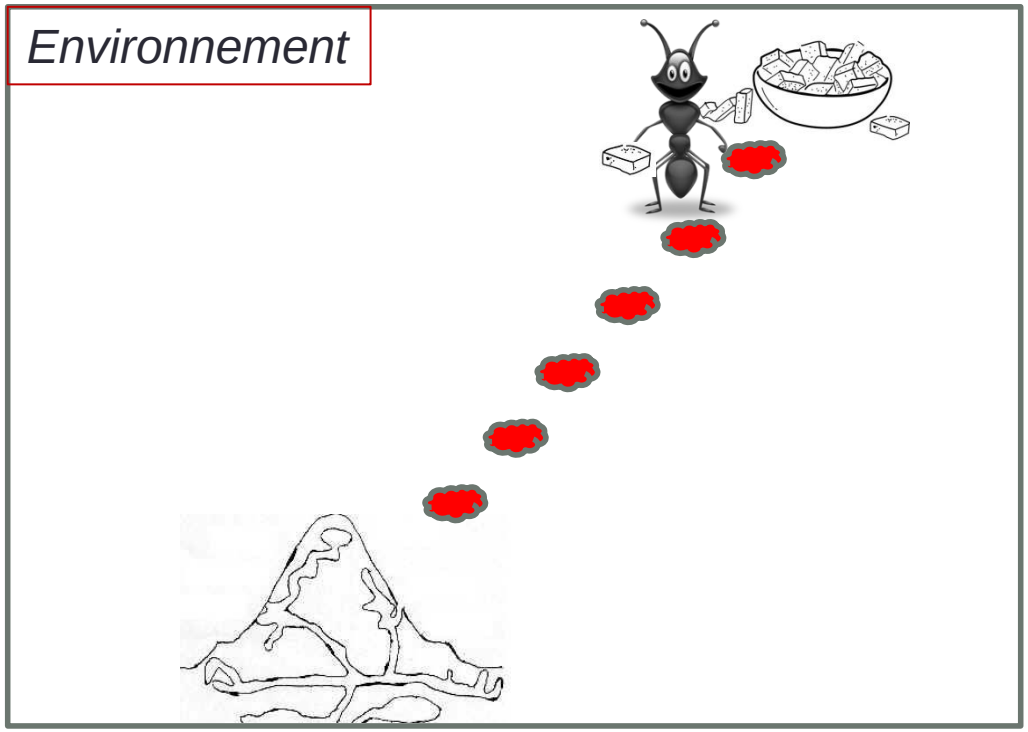




Règles de décision de l'agent :

- Explore aléatoirement
- **Si** non chargée **et** détecte source de nourriture **alors** ramasse nourriture **et** dépose phéromone

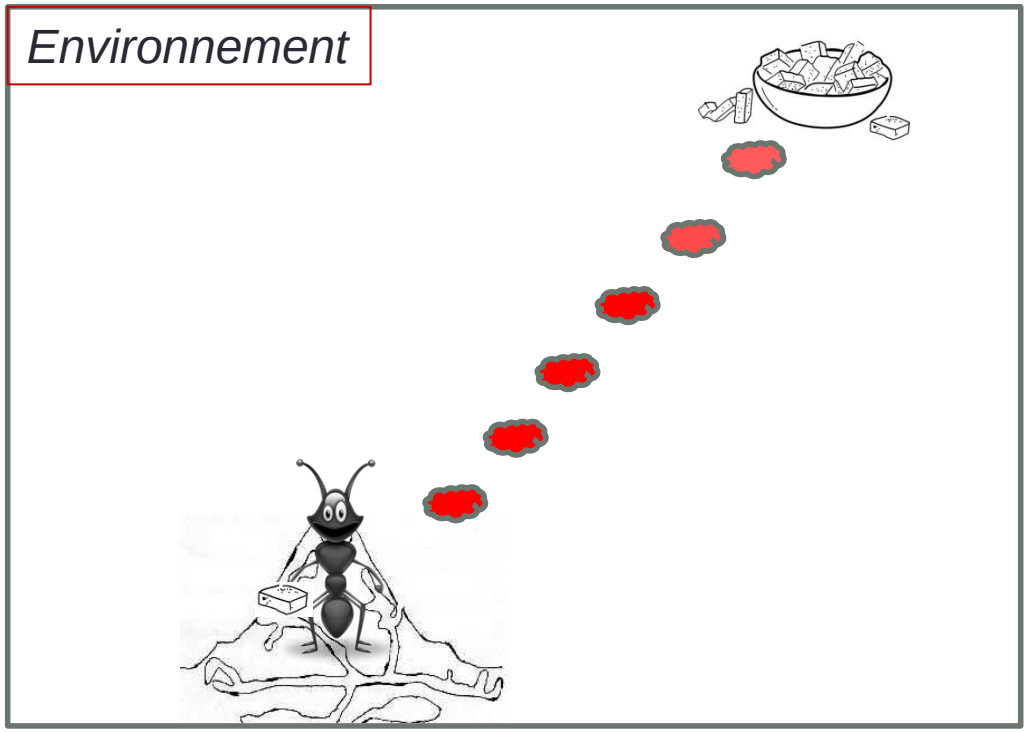
Exemple de simulation SMA
Les « autoroutes » de fourmis



Règles de décision de l'agent :

- Explore aléatoirement
- Si non chargée et détecte source de nourriture alors ramasse nourriture et dépose phéromone
- **Si** chargée **et** pas au nid **alors** va vers le nid **et** dépose phéromone

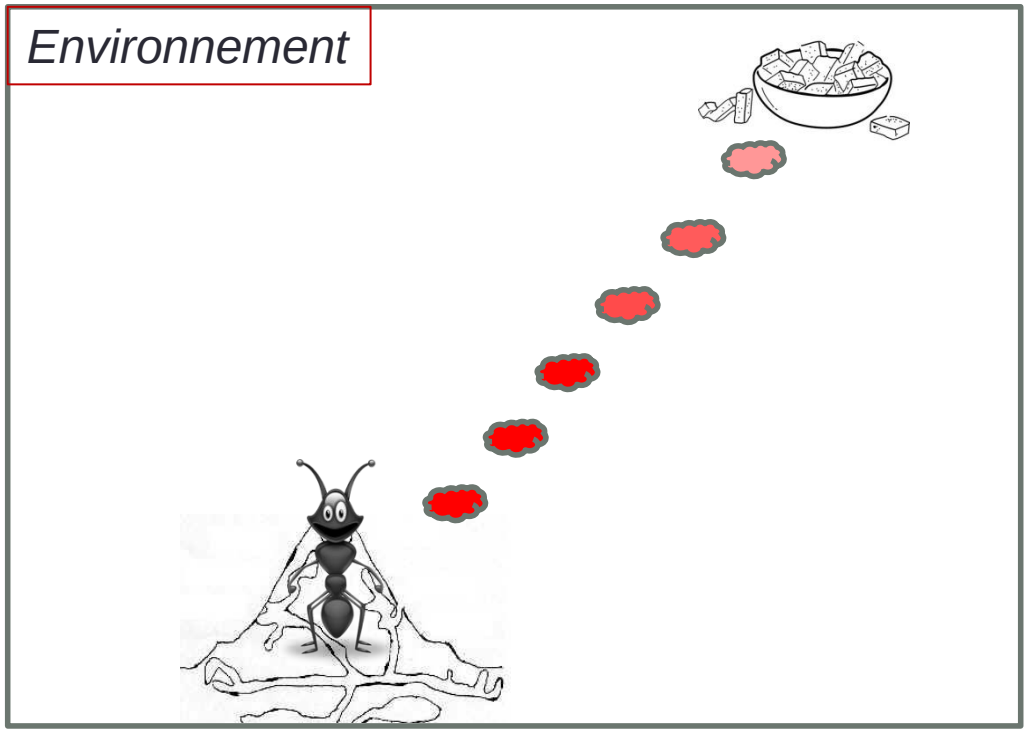
Exemple de simulation SMA
Les « autoroutes » de fourmis



Règles de décision de l'agent :

- Explore aléatoirement
- Si non chargée et détecte source de nourriture alors ramasse nourriture et dépose phéromone
- Si chargée et pas au nid alors va vers le nid et dépose phéromone
- **Si** chargée **et** au nid **alors** dépose nourriture

Exemple de simulation SMA
Les « autoroutes » de fourmis



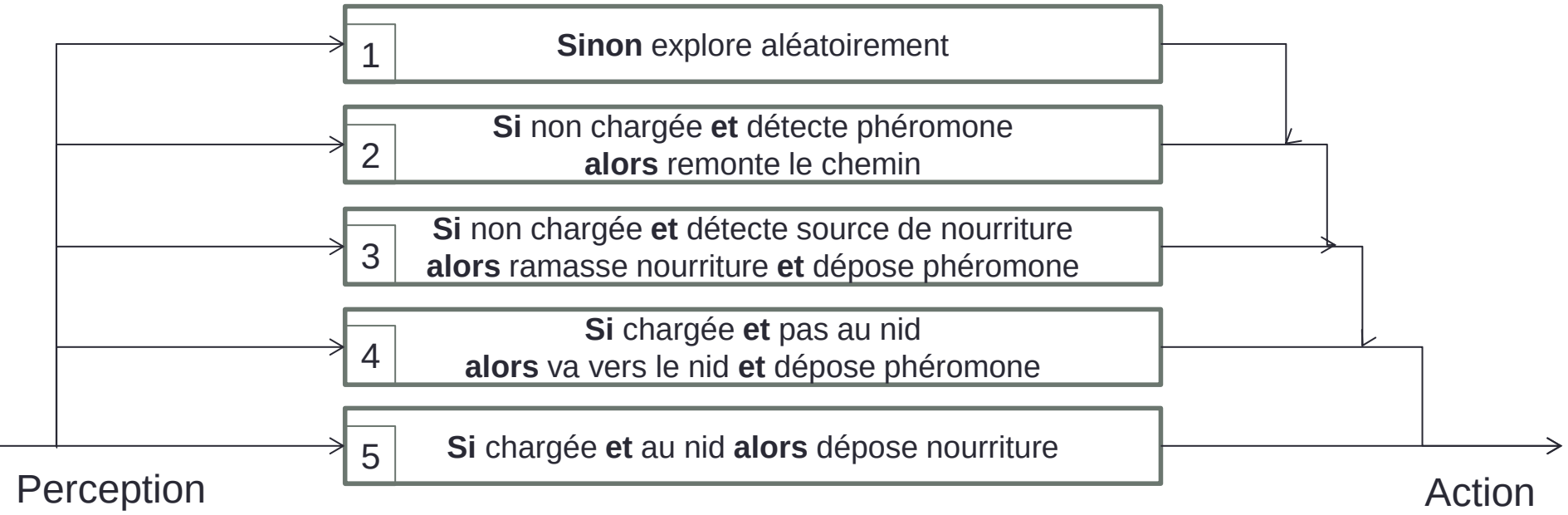
Règles de décision de l'agent :

- Explore aléatoirement
- Si non chargée et détecte source de nourriture alors ramasse nourriture et dépose phéromone
- Si chargée et pas au nid alors va vers le nid et dépose phéromone
- Si chargée et au nid alors dépose nourriture
- **Si** non chargée **et** détecte phéromone **alors** remonte le chemin

Exemple de simulation SMA
Les « autoroutes » de fourmis

Organisation des règles de décision

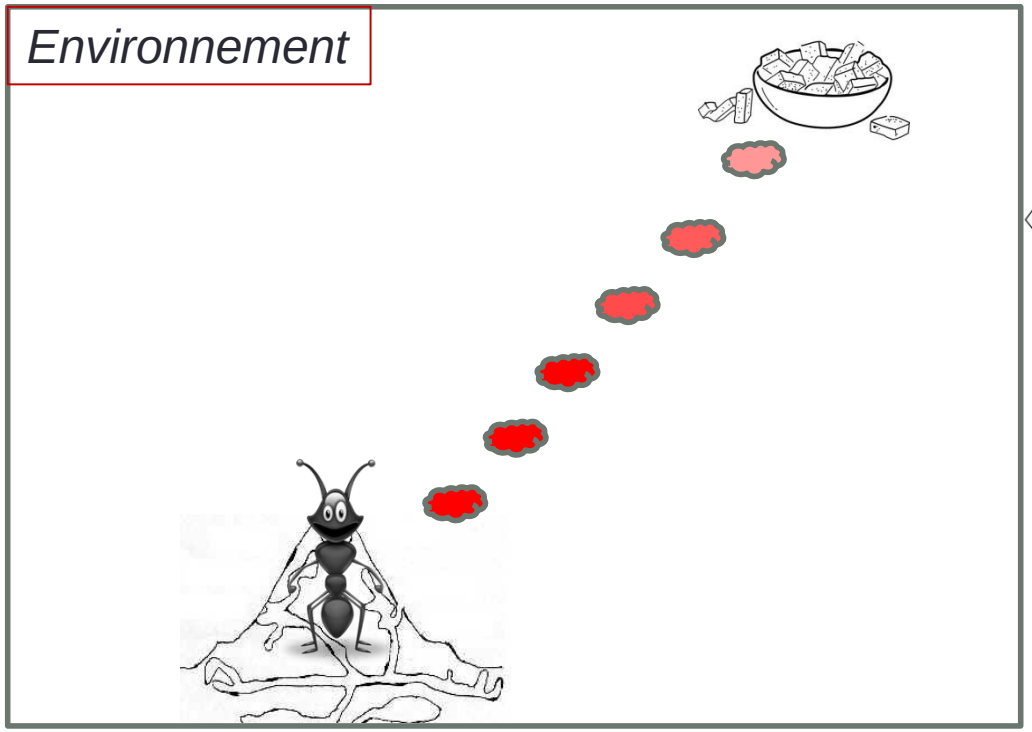
-> Architecture de **subsumption** (plus basse est la règle, plus haute sa priorité)

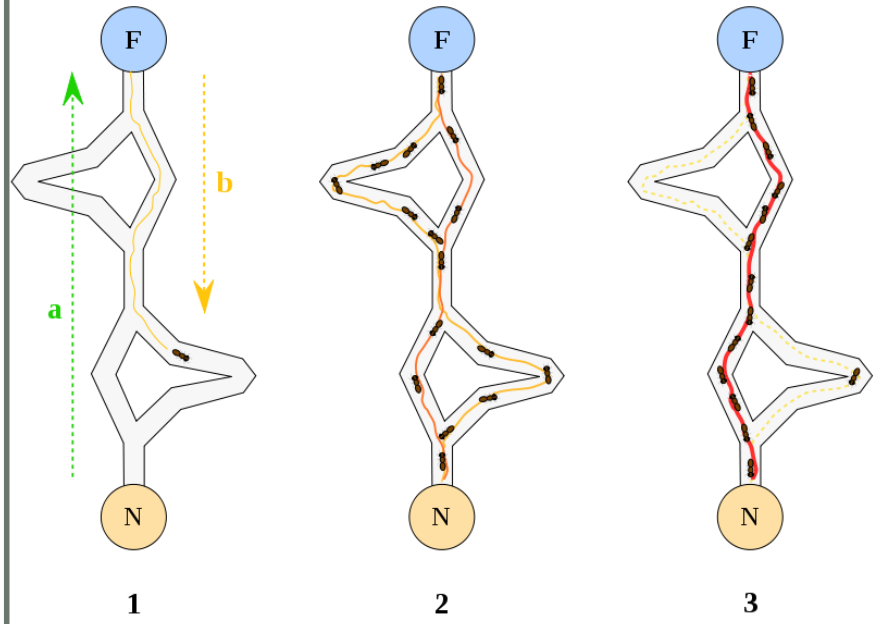


Ants.nlogo

Mise en évidence d'un **comportement d'auto-organisation**, la *stigmergie*, non déductible du seul comportement d'une fourmi
-> **Emergence**

Exemple de simulation SMA
Les « autoroutes » de fourmis





Wikipedia

Les « autoroutes » de fourmis
Qu'a-t-on fait ?

Focus sur l'environnement des agents : la fourmilière, les sources de nourriture

- Les agents et les interactions sont définis partir de l'environnement

- Modélisation **orientée environnement**

Il en existe d'autres !

- Modélisation orientée agents
- Modélisation orientée interactions
- Hybrides

Zoom sur les différentes approches sur une étude de cas

Etude de cas

Anna Dornhaus, Franziska Klügl, Christoph Oechslein et al., 2006
Benefits of recruitment in honey bees; effects of ecology and colony size in an individual-based model

Question scientifique :

Pourquoi certains insectes sociaux ont des systèmes de recrutements sophistiqués alors que d'autres espèces ne communiquent pas du tout sur la localisation des sources de nourritures ?

Mise en place d'un modèle à base d'agents

- Abeilles butineuses (*Apis mellifera*)
- Quantifier les bénéfices du recrutement selon :
  - différentes sources de nourritures (localisation, qualité, variabilité)
  - différentes tailles de colonies

Etude de cas
Modélisation orientée agents

Focus sur les agents, leur comportement et leur mécanismes de prise de décision

- Point de vue de l'agent !
- Interactions et environnement sont secondaires, et ajoutés selon le besoin

➢ **Stratégie**

1. Décrire de façon générale le comportement des agents : observation, littérature, experts du domaine
2. Catégoriser les agents : homogènes / hétérogènes ? Quels types d'agents ?
3. Décider de l'architecture des agents
4. Formaliser et implémenter le comportement des agents
5. Ajouter les aspects touchant aux interactions et à l'environnement si besoin
6. Simuler pour valider

Etude de cas
Modélisation orientée agents

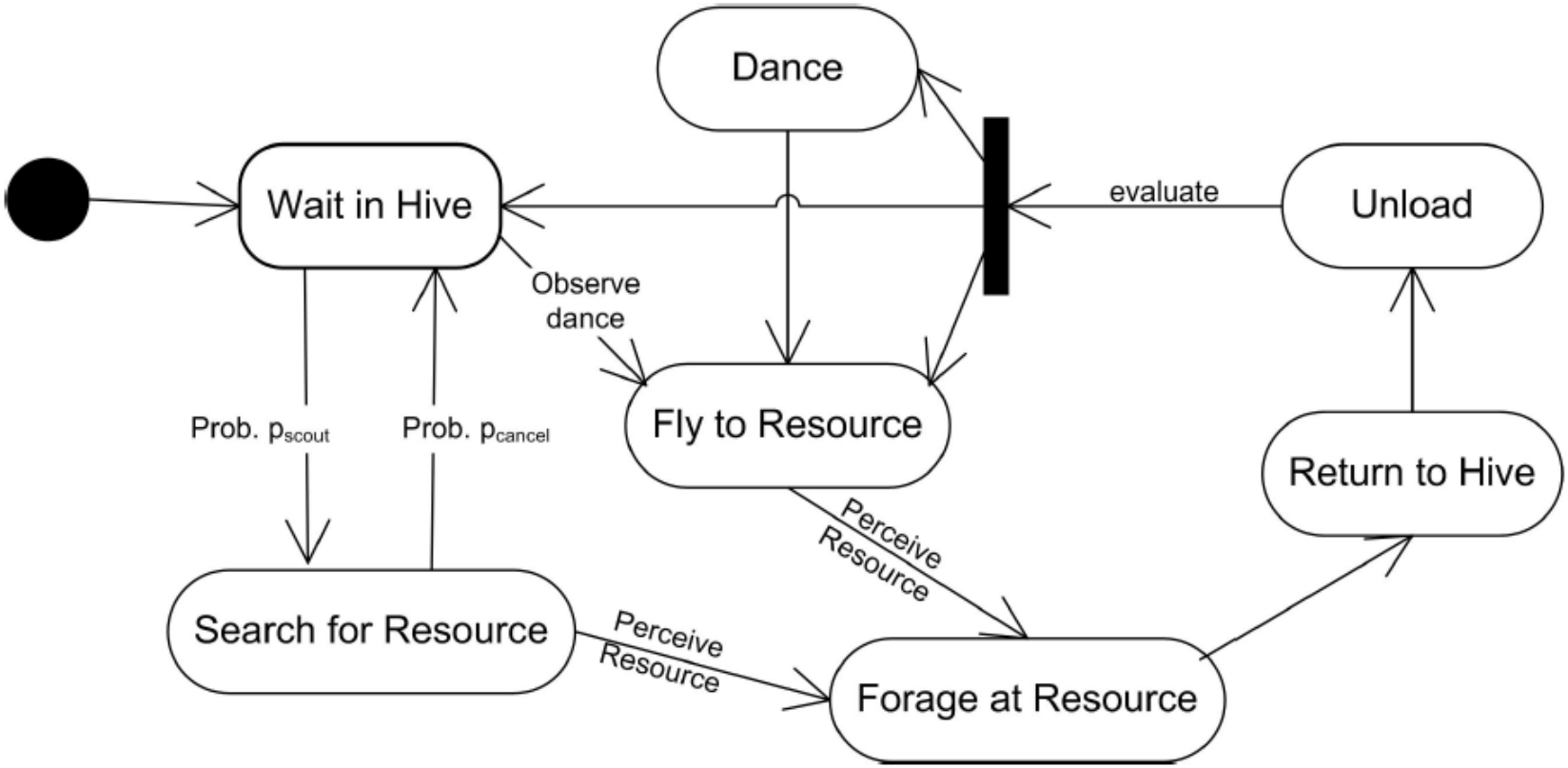
1. Comportement des agents : différentes activités
   Exploration, butinage, recrutement et attente de recrutement ..

2. Un seul type d'agent : l'abeille butineuse

3. Architecture simple, agents "réactifs"

4. Formalisation et implémentation : un diagramme d'activités

Etude de cas
Modélisation orientée agents



Environnement et interactions ?

Etude de cas
Modélisation orientée agents

1. Comportement des agents : différentes activités
   Exploration, butinage, recrutement et attente de recrutement ..

2. Un seul type d'agent : l'abeille butineuse

3. Architecture simple, agents "réactifs"

4. Formalisation et implémentation : un diagramme d'activités

5. Environnement et interactions :
   - une carte en 2 dimensions sur laquelle les agents se déplacent, explorent et découvrent les ressources, et butinent
   - la ruche : concept abstrait, un endroit pour stocker et une piste de danse
   - Danse : recrutement d'abeilles par une autre abeille

6. Simuler !

Modélisation orientée agents
Discussion

- Point de vue de l'agent
  - Comportementaliste
  - Jeux de rôle pour valider, découvrir …
- Intuitif
- Compliqué quand la validation n'est pas atteinte : Essais / erreurs
- Parfois non-trivial de trouver le comportement de l'agent -> intentionnalité ?
- Niveau de détails / granularité approprié ? Modèle le plus simple possible ?

Etude de cas
Modélisation orientée interactions

Focus sur les interactions, aux niveaux micro et macro

- Point de vue *d'en haut*
- Agents vus comme *des boites noires produisant les messages* -> Acteurs / Entités

➢ Stratégie

1. Identifier les acteurs / entités et les interactions entre eux
2. Décrire de façon générale les protocoles (acteurs + interactions) et leurs conditions, contraintes etc.
3. Ajouter les entités de l'environnement et dériver le comportement des agents pour qu'ils produisent les interactions recherchées
4. Implémenter et tester le comportement des agents
5. Simuler pour valider

Etude de cas
Modélisation orientée interactions

1. Identifier les acteurs / entités et les interactions entre eux

Interactions	Bees	Resources	Nectar Storage
Bees	Recruitment	Harvest	Unloading
Resources	Localization	-	-
Nectar Storage	Status Information	-	-

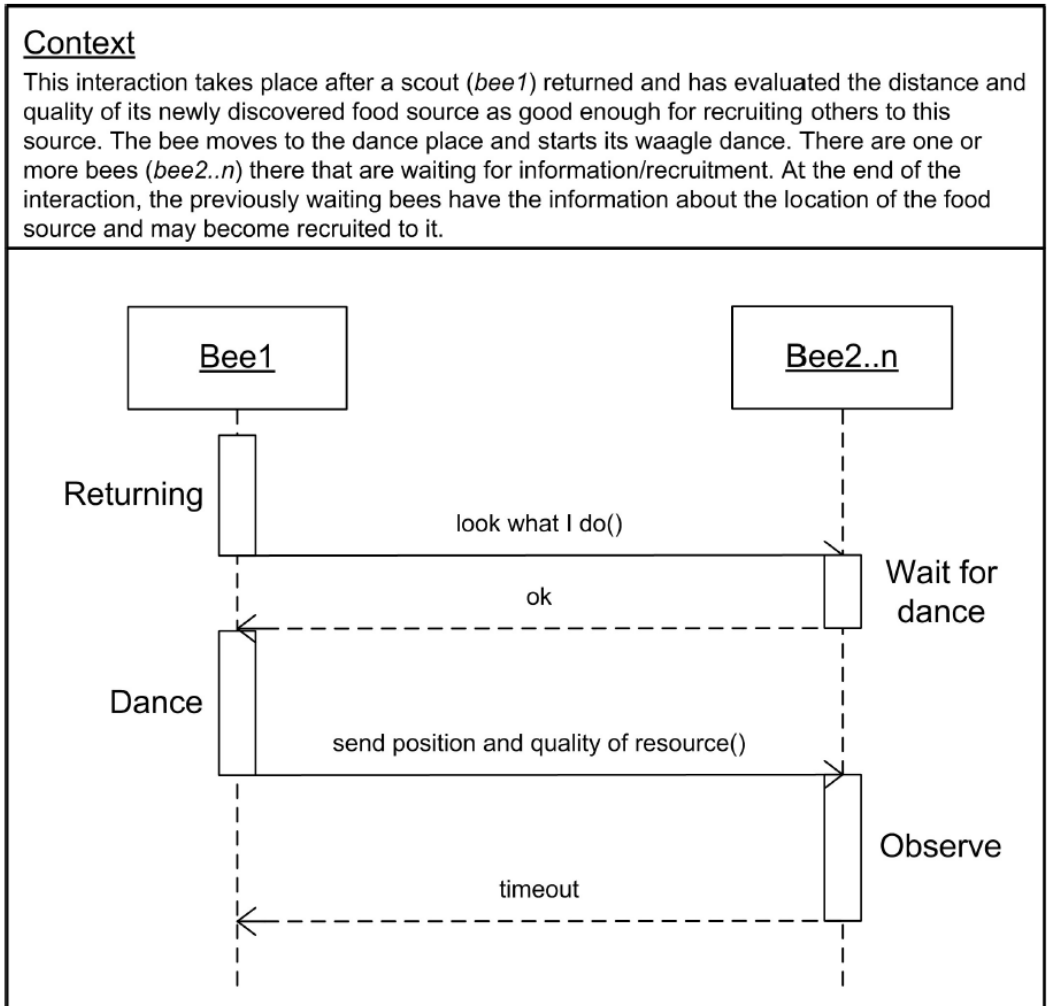
Table 8.1: Interaction table when choosing bees as agents.

Interactions	Scout	Forager	Reserve	Ressource	Nectar Storage
Scout	-	-	Recruitment (*)	Discovery (*)	-
Forager	-	-	-	Harvest	Unload
Reserve	Observe Dance (*)	-	-	-	-
Resource	Localization Information	Nectar	-	-	-
Nectar Storage	Status Information	Status Information	Status Information	-	-

Etude de cas
Modélisation orientée interactions

1. Identifier les acteurs / entités et les interactions entre eux

2. Protocoles : exemple du recrutement, avec son contexte et les différents acteurs



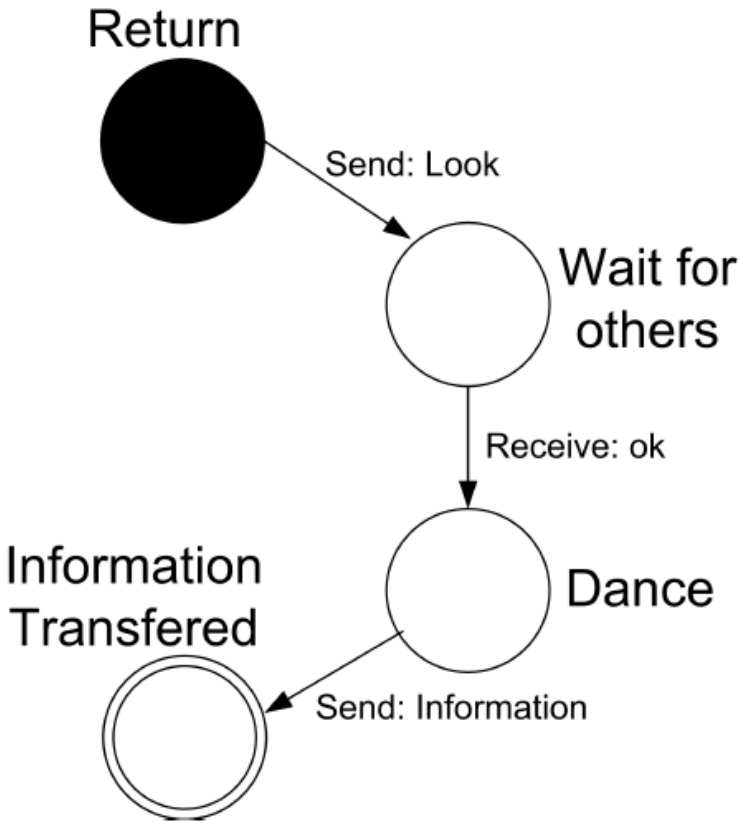
Etude de cas
Modélisation orientée interactions

1. Identifier les acteurs / entités et les interactions entre eux

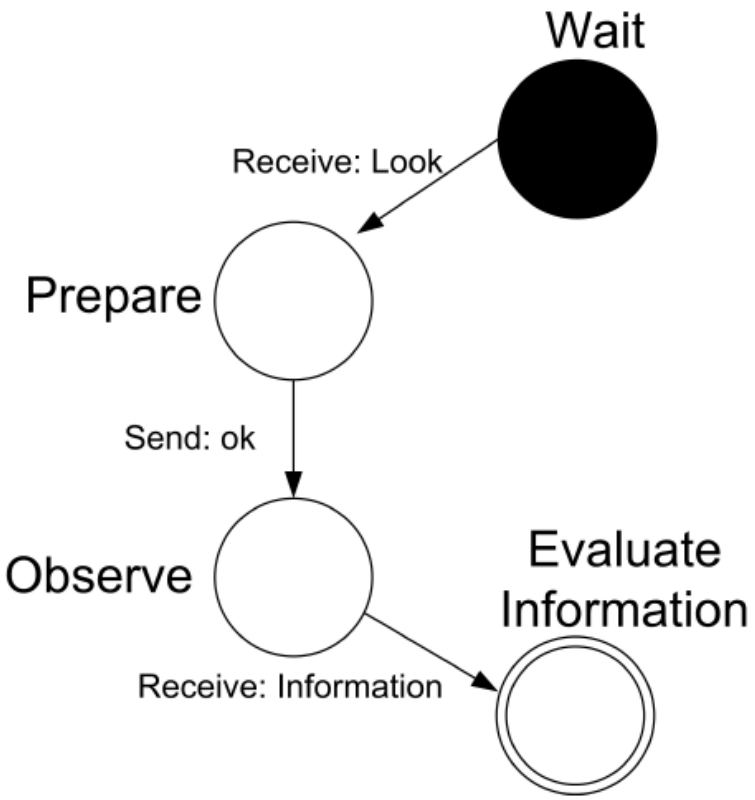
2. Protocoles : exemple du recrutement, avec son contexte et les différents acteurs

3. Dériver le comportement des agents
   - à partir de chaque protocole d'interaction
   - puis unifier pour avoir un seul comportement pour l'abeille
   - représentation, par exemple, à base de machine à états finis

Etude de cas
Modélisation orientée interactions

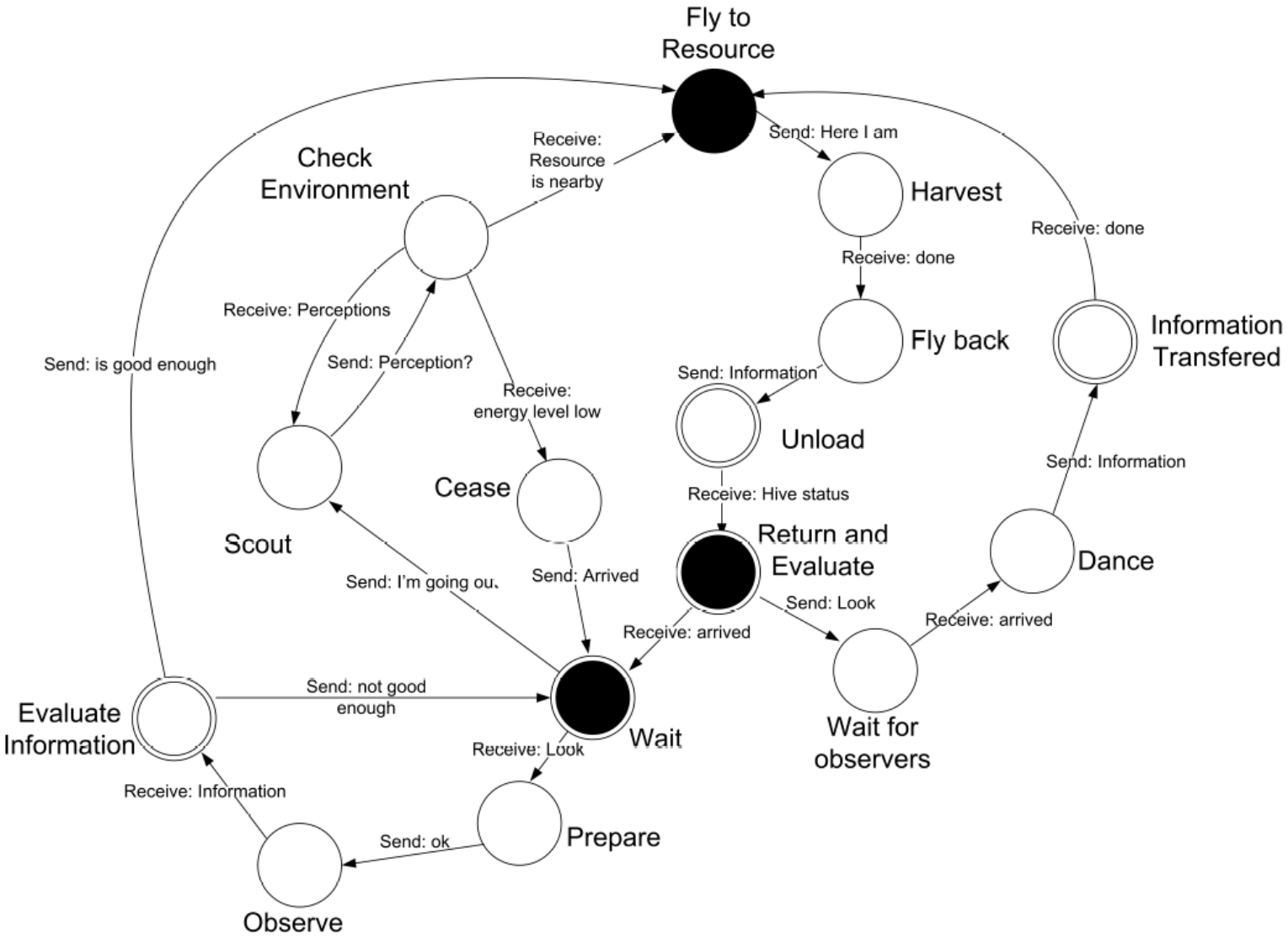


(a) Recruiter



(b) Recruitee

Etude de cas
Modélisation orientée interactions



Modélisation orientée interactions Discussion

- Dépendances / effets des interactions sur le comportement des agents représentées de manière explicite
- Problème avec des comportements pro-actifs, non-déclenchés par des messages externes
- Tout est agent !
- Stigmergie ? Représentation des phéromones ?
- Modélisation orientée **organisation**

Abstraction de l'organisation, définition de rôles pour les agents, et interactions selon les rôles …

Quand, comment on change de rôle ?

Etude de cas
Modélisation orientée environnement

Focus sur l'environnement dans lequel évoluent les agents

- Interactions et agents sont secondaires, et ajoutés selon le besoin

➢ Stratégie

1. Identifier les aspects pertinents de la partie du modèle qui représente l'environnement de l'agent
2. Déterminer les actions de l'agent et les réponses de l'environnement
3. Déterminer les informations de l'environnement qui doivent être perçues par l'agent pour sa prise de décision
4. Mettre en place une architecture d'agent apte à connecter ses perceptions et actions pour générer le comportement approprié.
5. Possibilité de mettre en place un mécanisme d'apprentissage automatique pour le comportement de l'agent, à base de récompense.
6. Implémenter le modèle d'environnement et le comportement des agents
7. Simuler, tester et analyser les résultats (possibles artefacts ?)

Etude de cas
Modélisation orientée environnement

1. Modèle d'environnement :
   - une carte en 2D (coordonnées)
   - une ruche, pour le stockage
   - des ressources distribuées sur la carte, contenant une certaine quantité de nectar

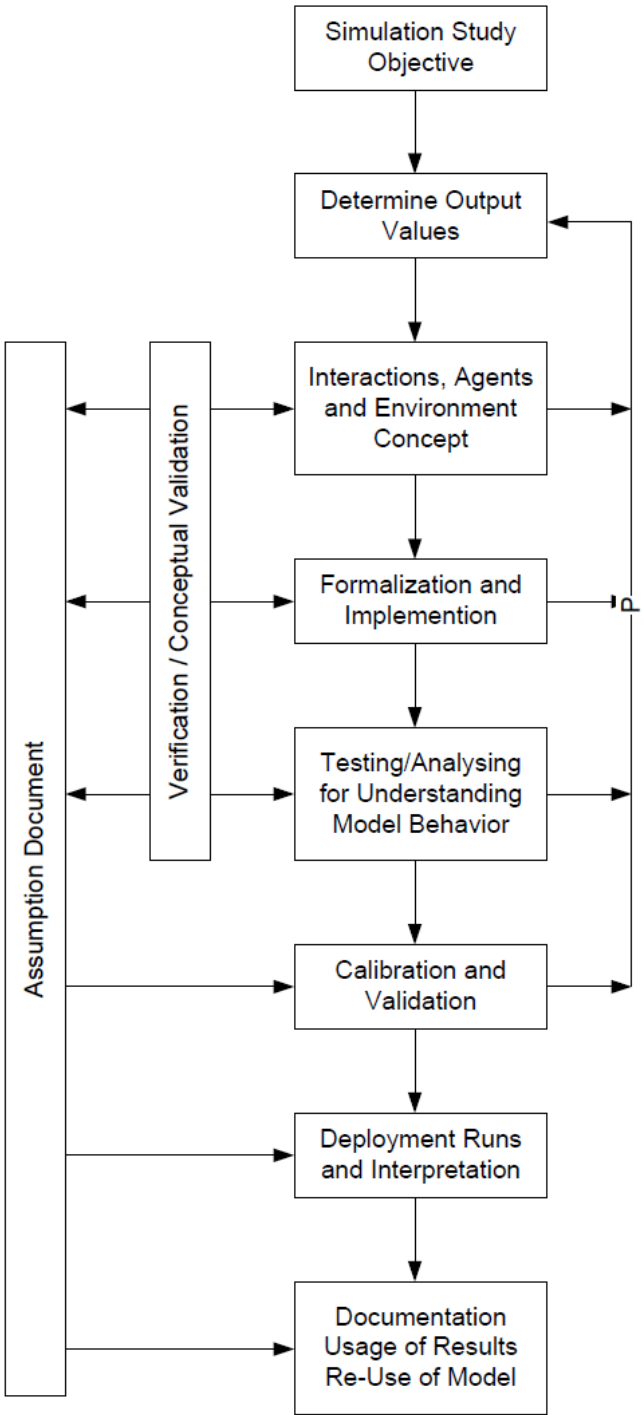
2. Configuration initiale :
   - ruche au centre de la carte
   - sources initialisées à des positions aléatoires
   - distribution normale du nectar

3. Perceptions et actions : existence d'une ressource (loin), sa position (proche), sa capacité (proche), les stocks, etc. ; voler vers une ressource, vers la ruche, charger/décharger nectar etc.

4. Approche à base de règles pour le comportement des agents

Modélisation orientée environnement Discussion

- Comme avec l'approche orientée interaction, problème avec des comportements pro-actifs, non-déclenchés par des stimulis externes
- Quel niveau de détails pour l'environnement ?
- Complexité des règles, parfois -> une autre architecture d'agent



Modélisation à base d'agents - Pour résumer

Les "bonnes" propriétés d'un modèle à base d'agents

- **Documentation** !

Hypothèses et affirmations scientifiques retenues, paramètres et leurs valeurs + sources

- Formalisation : subsomption, diagrammes d'états, tables d'interaction …

- Validité et reproductibilité
  - La **reproductibilité** des résultats est un prérequis majeur en sciences
  - Un modèle non-reproductible ne peut pas être réutilisé dans un but scientifique ou industriel
  - Un modèle va avec toute sa documentation, les données, détails d'implémentation etc.

- Simple (modèle minimal !), compréhensible, explorable

- Maintenable et extensible !

Modélisation à base d'agents
Challenges

- **Liens micro-macro**

  Les comportements des agents (au niveau micro) génèrent le comportement global du système (macro)
  - Liens entre les 2 niveaux souvent pas clairs, non-existants ou inconnus
  - On ne peut prédire exactement à l'avance quel va être le comportement macro (simulation)

- **Emergence et non-linéarité**

  Pire scénario micro-macro : pas de lien explicite entre les 2 niveaux
  - Non-linéarité des effets des comportements micro vers le macro
  - Boucles de feedback micro-macro

  -> Systèmes complexes ..

- Fragilité du modèle et **sensibilité aux paramètres**
  - Une (petite) modification peut compromettre le résultat de la simulation

    Valeur d'un paramètre, comportement, aspect technique

GameOfLife.nlogo

Modélisation à base d'agents
Challenges

- **Réglage des micro-règles** et risque de falsification

  Ne pas connaitre les paramètres du comportement de l'agent (micro)

  - Les calibrer, les modifier, dans le but d'obtenir ce que l'on souhaite au niveau macro

  -> Risque de comportements non-crédibles, qui produisent quand même l'effet recherché

- **Niveau de détails** et nombre d'hypothèses
  - Juste les hypothèses et les affirmations **nécessaires**

- **Taille et passage à l'échelle**
  - Un nombre d'agents minimum pour observer les résultats recherchés
  - Le modèle **et** le simulateur peuvent passer l'échelle

Le temps dans les simulations

- Simuler, c'est plonger un modèle dans le **temps**

- **La vitesse** à laquelle le temps s'écoule, **un paramètre** comme un autre dans la simulation
  - Ralentir le temps pour mieux observer
  - Accélérer le temps pour prédire
  - "En temps réel" pour intégrer l'humain
  - Au choix de l'utilisateur

- Temps **continu** ou **discret** ?
  - **Temps continu dans les modèles mathématiques** : l'intervalle entre 2 "actions" est arbitrairement petit
  - **Temps discret dans les modèles à base d'agents** : basé sur des événements (*event-based*) ou sur des pas de temps (*time-stepped*)

Le temps dans les simulations
Temps discret : time-step

- A chaque pas de temps, les agents font une action

- **Correspondance** entre temps virtuel et temps réel (règle de trois)
  - 1ms de temps virtuel = 1s de temps réel

- **Ordonnancement** et **synchronisation** !
  1. Version simple avec ordonnanceur
  2. Version multithread avec ordonnanceur

Implémentation des SMA

- Les langages de **programmation orientés objets** sont particulièrement adaptés aux agents : Java, C++ …

- Langages orientés **parallélisme et calculs distribués** (OpenCL)

A gérer :

- **Synchronisation** entre les processus, ressources partagées, etc.
- **Distribution** physique, si besoin !
- **Mécanismes de communication** entre les agents (format des langages, boites aux lettres …)
- **Visualisation** du comportement du système (messages échangés, "émergence", …)
- **Grand nombre** d'agents ?

Plateformes de simulation SMA

- De nombreuses **plateformes** existent !
  - Génériques : JADE, Jason, MadKit …
  - Pour la simulation : NetLogo, RePast, Gama, Flame , NetBioDyn …
  - Pour l'animation 3D (Massive), les marchés financiers (ATOM) …
  - Dépendantes d'un modèle (IODA - JEDI : modélisation orienté interactions)
  - …

- **Rôles**
  - **Gestion des cycles de vie des agents, *ordonnancement***
  - **Mécanismes de communication**
  - **Architecture distribuée**
  - **IHM de visualisation**
  - **Langages de programmation agents**

Agents et programmation Objet

- **Agent** : une extension du concept d'**objet** en programmation
  - **attributs**
  - capacités -> **méthodes *privées***

  + autonomie, comportement -> **threads** indépendants
  - exclusion mutuelle
  - ressources partagées
  - deadlock (interblocage)
  - livelock (famine)

  et distribution physique possible -> **application** indépendante

Agents et programmation Objet

- **Agent** : une extension du concept d'**objet** en programmation
  - **Attributs -> Attributs privés**

```
public class Fourmi{
    private boolean chargee;
    private Environnement env;
    private int x;
    private int y;

    public Fourmi(Environnement e){
        chargee=false;
        env=e;
        x=env.getNidX();
        y=env.getNidY();
    }
```

Agents et programmation Objet

- **Agent** : une extension du concept d'**objet** en programmation
  - Attributs -> Attributs privés
  - **Capacités de perception et d'action -> méthodes privées**

```
 * Choisis la prochaine position de la fourmi de façon alétoire,
private void explorer() {

 * @param detectee : la première marque detectée par la fourmi
private void remonterChemin(Marque detectee) {

 * La fourmi prend le plus court chemin pour revenir à son nid,
private void allerVersLeNid() {

 * Primitive de déplacement dans l'environnement, suivant l'échelle de la simulation
private void allerVers(int xd, int yd){
```

Agents et programmation Objet

- **Agent** : une extension du concept d'**objet** en programmation
  - Attributs -> Attributs privés
  - **Capacités de perception et d'action -> méthodes privées**

```
 * Choisis la prochaine position de la fourmi de façon alétoire,
private void explorer() {

 * @param detectee : la première marque detectée par la fourmi
private void remonterChemin(Marque detectee) {

 * La fourmi prend le plus court chemin pour revenir à son nid,
private void allerVersLeNid() {

 * Primitive de déplacement dans l'environnement, suivant l'échelle de la simulation
private void allerVers(int xd, int yd){
```

Communication ?

Agents et programmation Objet

- **Agent** : une extension du concept d'**objet** en programmation
  - Attributs -> Attributs privés
  - Capacités de perception et d'action -> méthodes privées
  - **Autonomie, comportement , ordonnancement ??**

Agents et programmation Objet Ordonnancement

1) **Version simple avec ordonnanceur**

- Une méthode publique *vivre*
- **Un ordonnanceur** (*scheduler*) pour faire *vivre* les agents, **dans un certain ordre**, à chaque **tick**

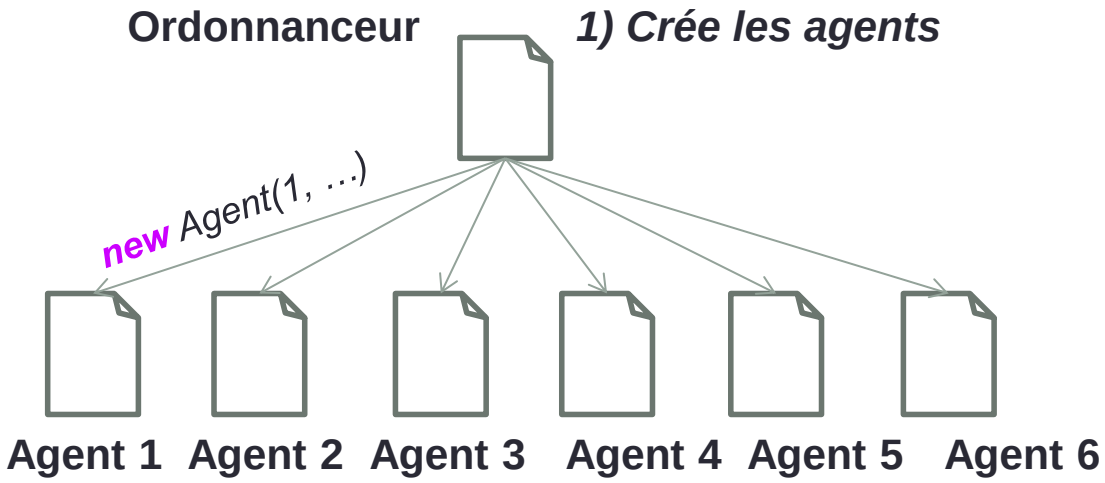
- **<u>Conserver l'état du monde</u> à *t***

```
public void vivre() {
    if((isChargee())&&(env.auNid(x, y))){
        env.deposeNourriture();
        chargee=false;
    }
    else if((isChargee())&&(!env.auNid(x, y))){
        env.deposerMarque(x, y);
        allerVersLeNid();
    }
    else if((!isChargee())&&(env.detecteMarque(x, y)!=null)){
        remonterChemin();
    }
    else if((!isChargee())&&(env.detecteNourriture(x, y))){
        env.ramasseNourriture();
        chargee=true;
        env.deposerMarque(x, y);
    }
    else{
        explorer();
    }

}
```

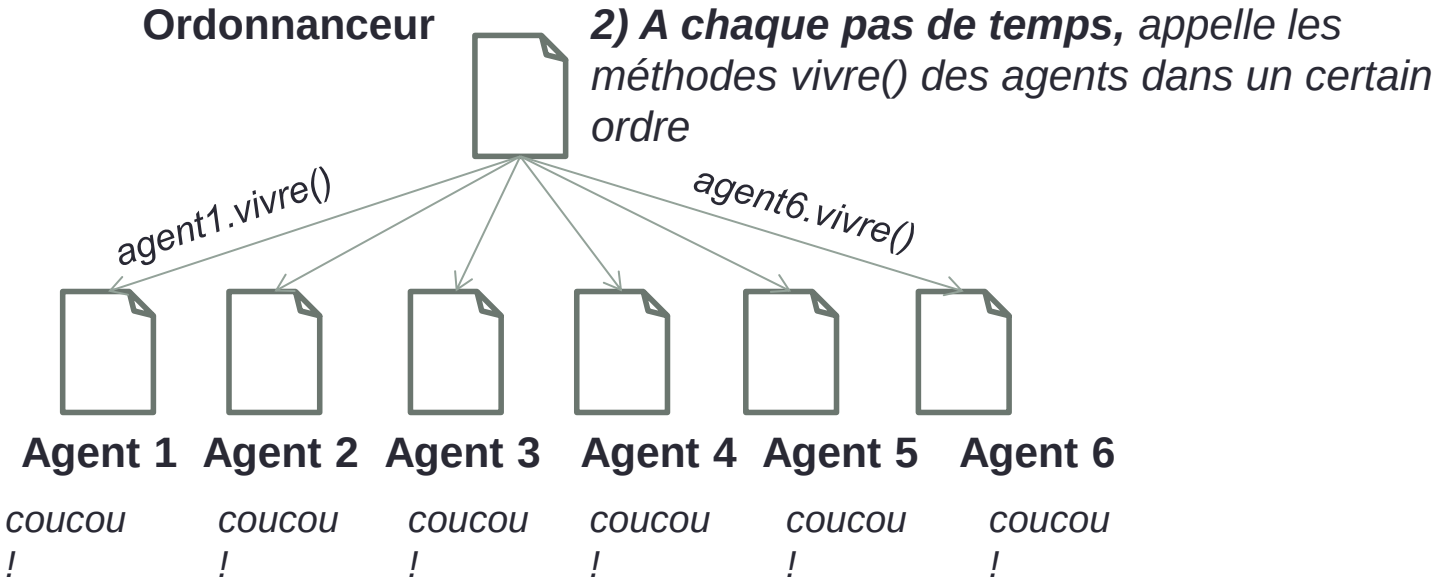
Agents et programmation Objet Ordonnancement

Version simple avec ordonnanceur



Agents et programmation Objet
Ordonnancement

Version simple avec ordonnanceur

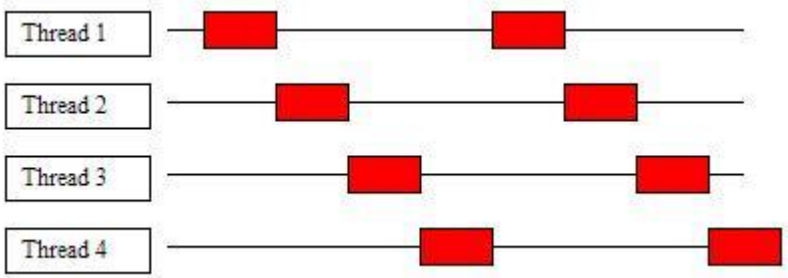


- *Ordonnancement simple : toujours dans le même ordre*
- *Ordonnancement chaotique : ordre aléatoire à chaque pas de temps*

Agents et programmation Objet Ordonnancement

2) **Version multithread** avec **ordonnanceur**

- **Thread =** traitement, tâche
  - != processus (espaces virtuels indépendants)
  - **Exécution en parallèle ssi** le nombre de threads <= nombre de processeurs
  - Sinon, **exécution en temps partagé** sur un (ou plusieurs) processeurs
    -> **ordonnancement automatique**



Agents et programmation Objet Ordonnancement

Cycle de vie d'un Thread

1. Nouveau : état initial après l'instanciation du thread
2. Exécutable (Thread actif) : après l'appel à la méthode *start()*, et tant que le Thread n'est pas sorti de sa méthode *run()*
3. En attente : n'exécute aucun traitement et ne consomme aucune ressource CPU
   - méthode *sleep(temps en millisecondes)*
   - méthode *wait()*
   - accès à une ressource bloquante (flux, accès en base de données, etc.)
   - accès à une instance sur laquelle un verrou a été posé
4. Mort : le Thread est sorti de sa méthode *run()*, soit de manière naturelle, soit de manière subite (exception)

Agents et programmation Objet Ordonnancement

Java : la classe Thread

- Implémente l'interface Runnable
- Constructeurs : Thread() ; Thread(Runnable r); …
- Quelques méthodes :

run()	*setDaemon(boolean b)*
start(), sleep(long ms)	*yield() - statique*
interrupt(), isInterrupted()	*join() - statique*
setPriority(int p)	

- Pour le reste : http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html

Agents et programmation Objet Ordonnancement

Créer et lancer un Thread :

- La classe Agent doit, soit

Implémenter l'interface Runnable, et définir la méthode *run()*

```
public class Agent implements Runnable {

@Override
public void run() {
    int i = 0;
    for (i = 0; i < 10; i++) {
                System.out.println("" + i);
     }
}
```

```
Thread th=new Thread(new Agent());
th.start();
```

Etendre la classe Thread, et surcharger (redéfinir) la méthode *run()*

```
public class Agent extends Thread {

public void run() {
    int i = 0;
    for (i = 0; i < 10; i++) {
                System.out.println("" + i);
     }
}
```

```
Agent ag=new Agent();
ag.start();
```

Agents et programmation Objet Ordonnancement

Arrêter un Thread :

- Méthodes *stop()* et *suspend()* **dépréciées depuis Java 1.2**
- Utiliser un booléen

```
public void run() {
  while (active)){
      // traitements ...
   }
}

 private void finish() {
active = false;
}
```

Agents et programmation Objet Ordonnancement

Arrêter un Thread :

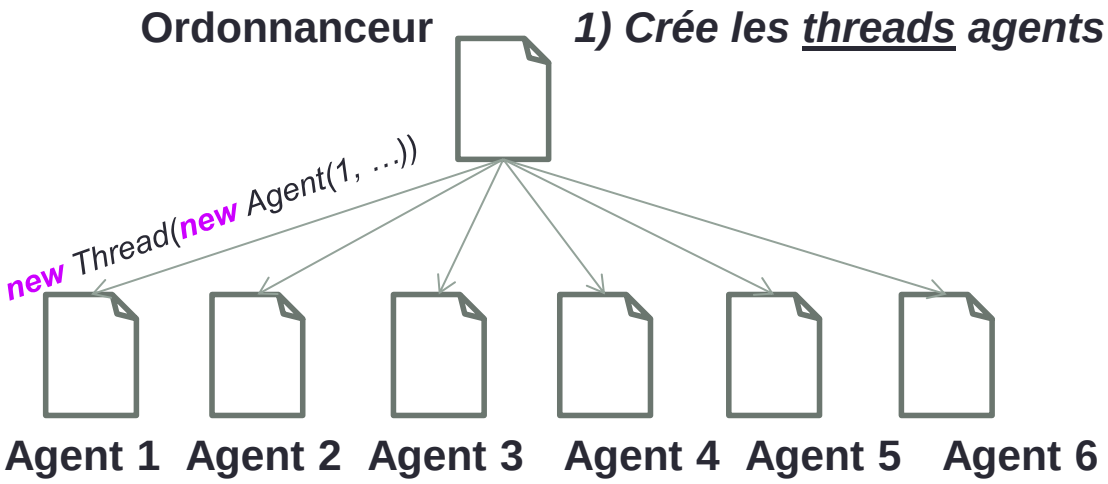
- Méthodes *stop()* et *suspend()* **dépréciées depuis Java 1.2**
- Utiliser un booléen
- Utiliser les méthodes d'interruption

```
public void run() {
 try {
          while (!Thread.currentThread().isInterrupted()){
                // traitements ...
                }
       } catch (InterruptedException e) {
                // nous avons été interrompu
                Thread.currentThread().interrupted() ; }
}

 private void finish() {
                Thread.currentThread().interrupt() ;
}
```

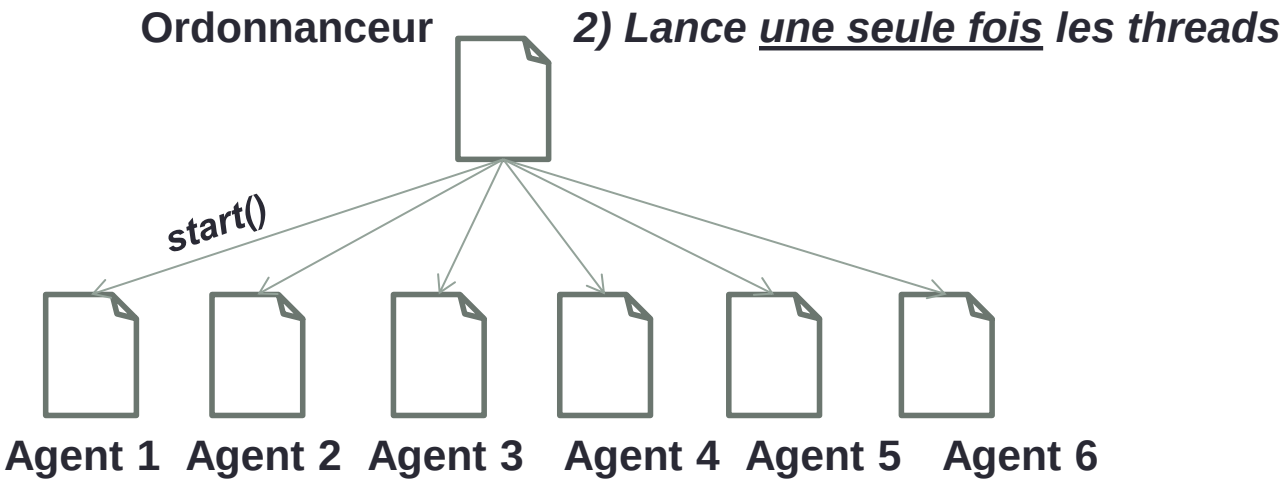
Agents et programmation Objet
Ordonnancement

Version multithread avec **ordonnanceur**



Agents et programmation Objet
Ordonnancement

Version multithread avec **ordonnanceur**



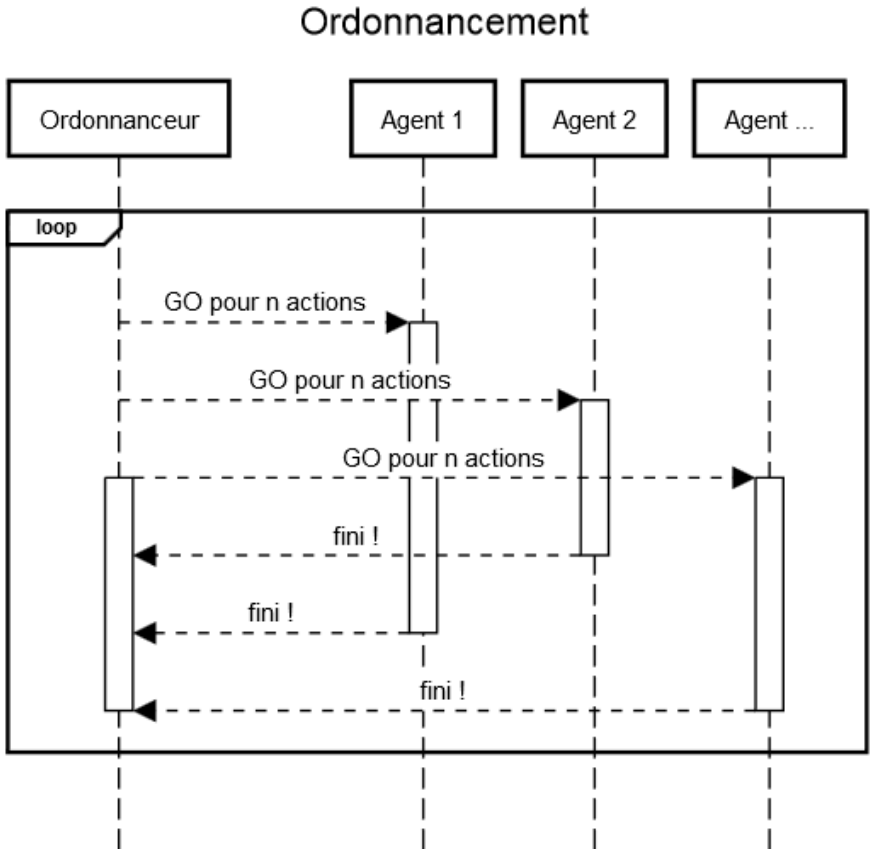
Tant qu'un agent est en vie, il suit son cycle de perception – décision - action : sa méthode *run()*

Question : comment assurer maintenant que chaque agent "vit" à la même vitesse, le même nombre de fois ?

Agents et programmation Objet
Ordonnancement

Question : comment assurer maintenant que chaque agent "vit" à la même vitesse, le même nombre de fois ?

Par exemple :



Agents et programmation Objet

Ressources, données partagées

- Synchroniser les accès aux données partagées !!

Par exemple :

Une classe Environnement

```
public void ramasseNourriture(){
    if(nourriture.getQuantite()>0){
        nourriture.setQuantite(nourriture.getQuantite()-1);
        if(nourriture.getQuantite()==0) aSupprimer.add(nourriture);
    }
}
```

Quantité = 1

Des instances de la classe Fourmi (agents)


Un Thread par fourmi

Agents et programmation Objet

Ressources, données partagées

- Synchroniser les accès aux données partagées !!

ramasseNourriture()
Quantité>0 ? Oui

ramasseNourriture()
Quantité>0 ? Oui !

Quantité = 0 -1

```
public synchronized void ramasseNourriture(){
    if(nourriture.getQuantite()>0){
        nourriture.setQuantite(nourriture.getQuantite()-1);
        if(nourriture.getQuantite()==0) aSupprimer.add(nourriture);
    }
}
```

Agents et programmation Objet

Deadlock

- Une mauvaise gestion de la synchronisation entre blocs peut mener à une situation de blocage total :

T1 se trouve dans un bloc synchronisé B1, et est en attente à l'entrée d'un autre bloc synchronisé B2.

T2 se trouve dans le bloc synchronisé B2, et est en attente à l'entrée d'un autre bloc synchronisé B1.

- Eviter d'**imbriquer les appels synchronisés**
- **Documenter**
- Utiliser les outils de **java.util.concurrent** pour synchroniser des portions de code

Agents et programmation Objet

Famine

- Lorsqu'un algorithme ne garantit pas à tous les threads souhaitant accéder à une section critique une probabilité non nulle d'y parvenir en un temps fini
  - ex : *synchronized* ne fonctionne pas en FIFO !

Livelock

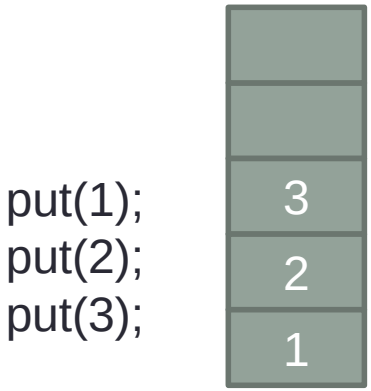
- Un thread agit en fonction d'un autre .. qui agit en fonction de ce thread, ou d'un autre
- Les threads ne sont pas bloqués, ils sont trop occupés !
  - ex : 2 personnes trop polies dans un couloir

Agents et programmation Objet

Le package java.util.concurrent

- Introduit dans Java 5
- Propose des outils (classes) pour faciliter le développement de systèmes concurrents et multithreads:
  - **BlockingQueue** (interface) : file FIFO synchronisée

-> Méthodes *put(Object)* et *take*() **bloquantes**

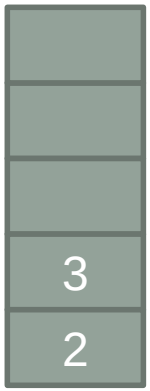


Agents et programmation Objet

Le package java.util.concurrent

- Introduit dans Java 5
- Propose des outils (classes) pour faciliter le développement de systèmes concurrents et multithreads:
  - **BlockingQueue** (interface) : file FIFO synchronisée

-> Méthodes *put(Object)* et *take*() **bloquantes**



take() ->
1

Agents et programmation Objet

Le package java.util.concurrent

- **ConcurrentMap** (interface) : Map synchronisée

  putIfAbsent(K key, V value), remove(Object key, Object value), …

- **Lock** (interface) : verrous sur une section de code

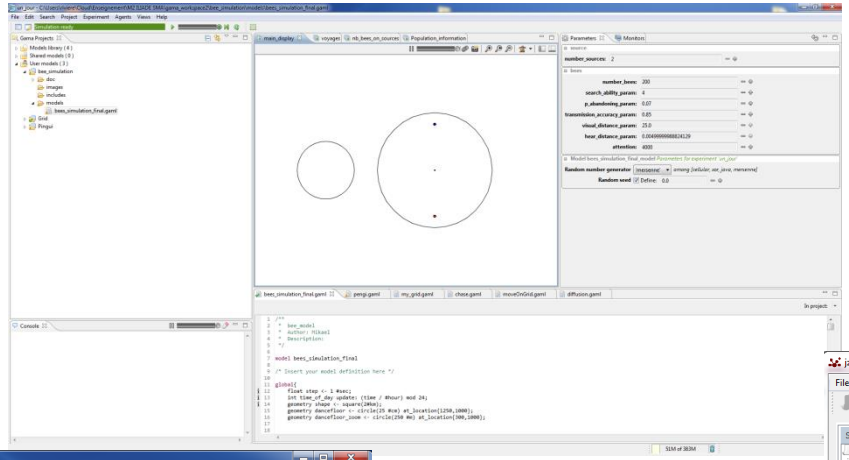
```
Lock lock = new ReentrantLock();
lock.lock();
//section critique
 lock.unlock();
```

Plan

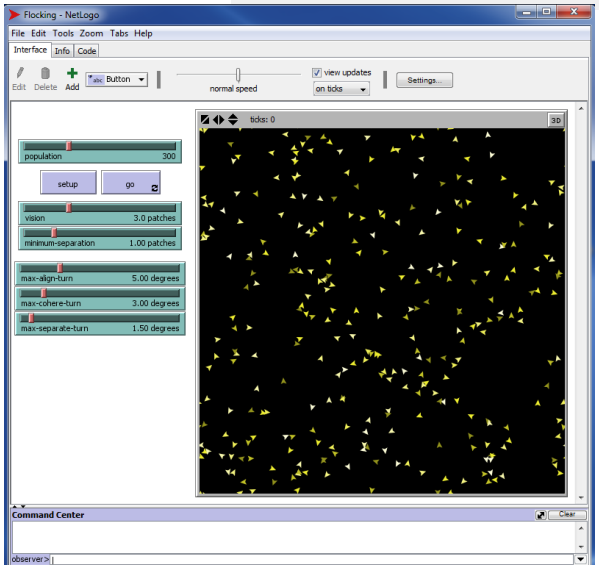
1. Introduction aux systèmes complexes
2. Modélisation et simulation
3. Les SMA pour la simulation et la modélisation de systèmes complexes
4. Visualiser et interagir avec des simulations à base d'agents
   1. Manipulation et visualisation des simulations : IHM de plateformes
   2. Simulation participative, Apprentissage, EVAH, Interactions intuitives

Manipulation et visualisation des simulations
IHM

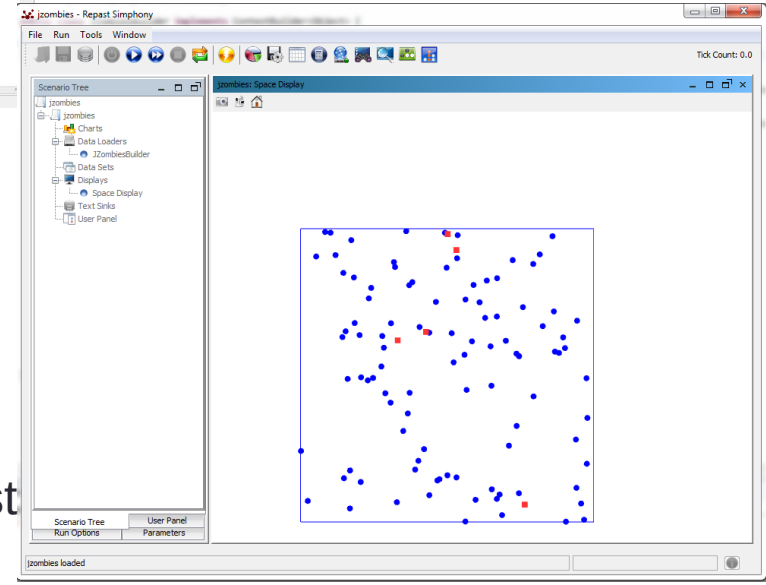
Des **IHM** variées selon les plateformes pour valider et expérimenter les modèles



Gama



NetLogo



Repast

Manipulation et visualisation des simulations
IHM

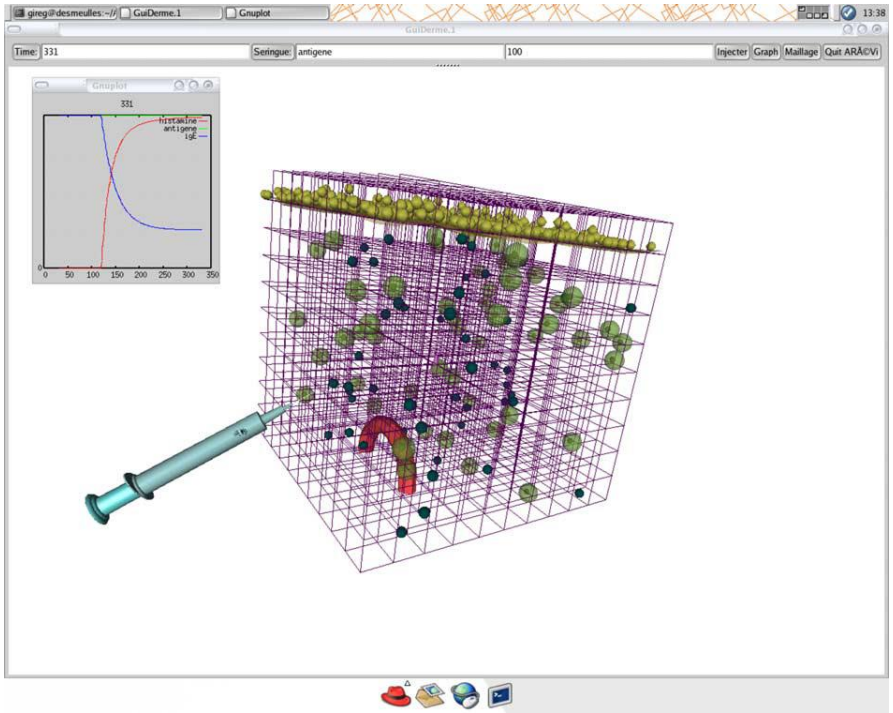
Visualisation	Interactions
2D / 3D selon les plateformes / les modèles	**Avec la simulation** • Lancer la simulation • Contrôler le temps de la simulation • Modifier l'environnement (état initial ou pendant le déroulement)
Au niveau **micro** : Comportement **des agents** • Mouvements • Prises de décision • Interactions (messages …)	
Au niveau **macro** : Comportement **du système** • Courbes (évolution de la population …) • Phénomènes émergents • Auto-organisation	**Avec les agents** • Ajout / Suppression / Déplacement d'agents -> "*Point and click*" • Envoi de messages

Manipulation et visualisation des simulations
IHM

Mais aussi

Métaphore de l'interaction -> IHM à destination de non-informaticiens

Par ex. Le Laboratoire Virtuel [Desmeulles, 2006]



Simulations participatives

Simulations multi-agents participatives [Guyot, 2006]

Une simulation avec laquelle les humains interagissent directement en contrôlant un des agents du système -> **simulations sociales**

- Des expériences menées en laboratoires ou à travers le réseau Internet,
- Avec des participants humains et qui s'inscrivent dans une démarche multi-agents

Buts

- Apprentissage et entrainement
- Validation du modèle
- Aide à la décision

Les moyens d'interaction sont les mêmes que ceux des agents du système

En immersion -> EVAH et Réalité virtuelle

Apprentissage, EVAH, Interactions intuitives
Réalité virtuelle

La **réalité virtuelle** comme vecteur d'interaction et de visualisation avec la simulation

Environnements Virtuels pour l'Apprentissage Humain

MASCARET [Querrec, 2011]

Apprentissage, EVAH, Interactions intuitives
Réalité virtuelle

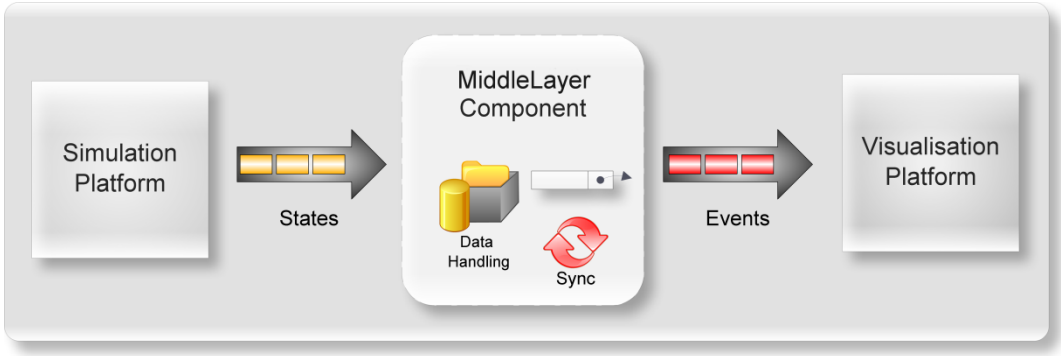
Visualisation	Interactions
3D, Réalité virtuelle, tangible, haptique	**Avec les agents** • L'humain est un **avatar** perçut par les agents • Interactions intuitives : langage naturel, gestes
Au niveau **micro** : Comportement **des agents** • Mouvements • Interactions (voix, gestes) Environnement • « Physique » • Augmenté avec des indications du système d'apprentissage	

Apprentissage, EVAH, Interactions intuitives Réalité virtuelle

Challenges

- Correspondances Simulation - RV
  - Agents -> Objets 3D
  - Comportements / Etats des agents -> Animations
- Dynamiques de l'environnement (température, brouillard, signaux …)
- Interactions entre les agents
- Synchronisation des temps simulateur / plateforme de visualisation
- Interpolation des comportements
- Configuration du monde virtuel

- Interactions
  humain - simulation !



Louloudi and Klügl, 2012

Bibliographie / Références

Systèmes complexes

- Melanie Mitchell, **Complexity : a guided tour**, 2009.
- **Complexity: Life at the Edge of Chaos.** Roger Lewin. Dent
- **Les Systèmes complexes.** Hervé P. Zwirn. Odile Jacob
- Institut des systèmes complexes https://iscpif.fr/
- Santa Fe Institute: http://www.santafe.edu/

Systèmes multi-agents :

- M. Wooldridge. ***An Introduction to MultiAgent Systems***, Wiley, 2002
- J. Ferber. ***Les systèmes Multi-Agents***, InterEditions, 1995
- Kravari, Kalliopi and Bassiliades, Nick. ***A Survey of Agent Platforms***, Journal of Artificial Societies and Social Simulation 18 (1) 11, 2015.
- Macal, Charles, and Michael North, 2005**, Tutorial on Agent-based Modeling and Simulation**,Proc. 2005 Winter Simulation Conference, M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds., Orlando, FL, Dec. 4-7, pp. 2-15,

Bibliographie / Références

Visualisation et interactions avec les SMA

- Ronan Querrec. **Apprentissage de procédures en environnements virtuels** . HdR, 2011
- Athanasia Louloudi, Franziska Klügl-Frohnmeyer. **A New Framework For Coupling Agent-Based Simulation And Immersive Visualisation**. In ECMS, 2012.
- Paul Guyot. **Simulations multi-agents participatives: Faire interagir agents et humains pour modéliser, explorer et reproduire les comportements collectifs.** Thèse, 2006
- Gireg Desmeulles et al. **The virtual reality applied to biology understanding: The in virtuo experimentation.** In Expert Systems with Applications, 2006.