Université de Msila

Faculté des mathématiques et de l'informatique

Département d'informatique

1ère année Master IA

Réalisation informatique des simulations à base d'agents

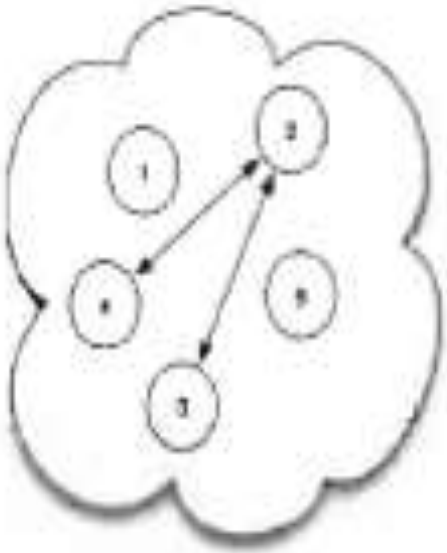
Gestion du temps

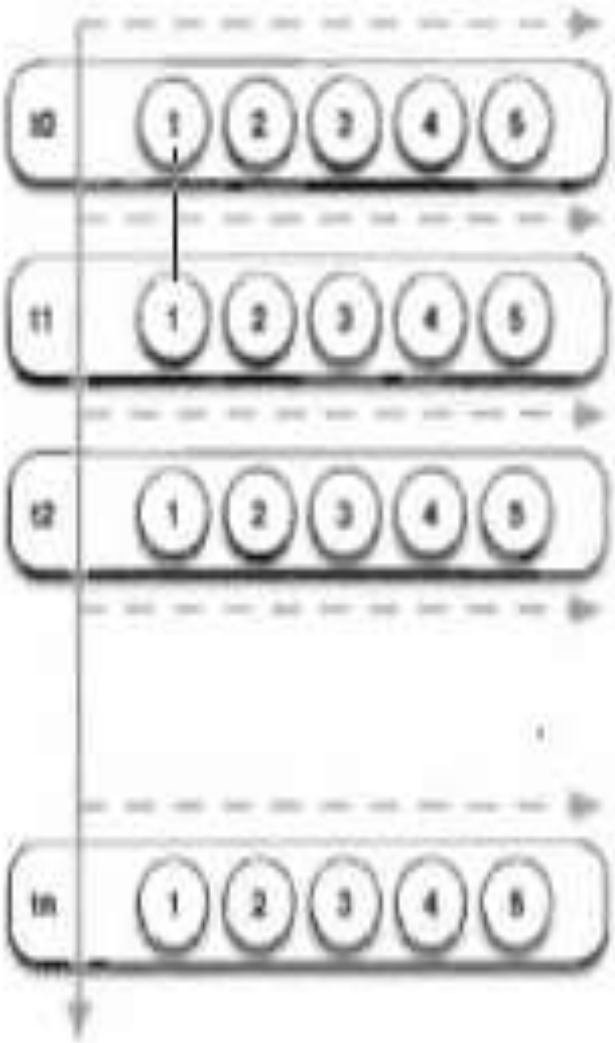
- Le méta-modèle présenté est statique, c'est-à-dire qu'il présente l'organisation des concepts permettant de réaliser une modélisation à base d'agents d'un système de référence, mais il n'aborde pas l'aspect dynamique des modèles
- Modéliser le concept de temporalité est un problème difficile. Il existe aujourd'hui deux grandes techniques de gestion du temps dans une simulation : celles qui sont dirigées par les événements et celles qui sont dirigées par une horloge.
- Les horloges proposent de discrétiser le temps, c' est-à-dire de le segmenter en un ensemble de pas de temps de même durée. Les changements d'états ne s'opèrent que lors du passage d'une date à une autre. Nous pouvons dire que le temps est indépendant de l'entité : lors de la transition d'une date à une autre de ce *temps discret* , il faut« réveiller » toutes les entités afin de procéder aux changements d'états spécifiés
- Dans une simulation avec une gestion du temps orientée par les événements, le système propose un échéancier commun, sur un *temps* pouvant être conceptuellement *continu,* dans lequel les entités peuvent inscrire la date à laquelle leurs changements d'états surviendront. Le temps va donc progresser d' une date d'occurrence d'événement à une autre.

Ordonnancement des agents

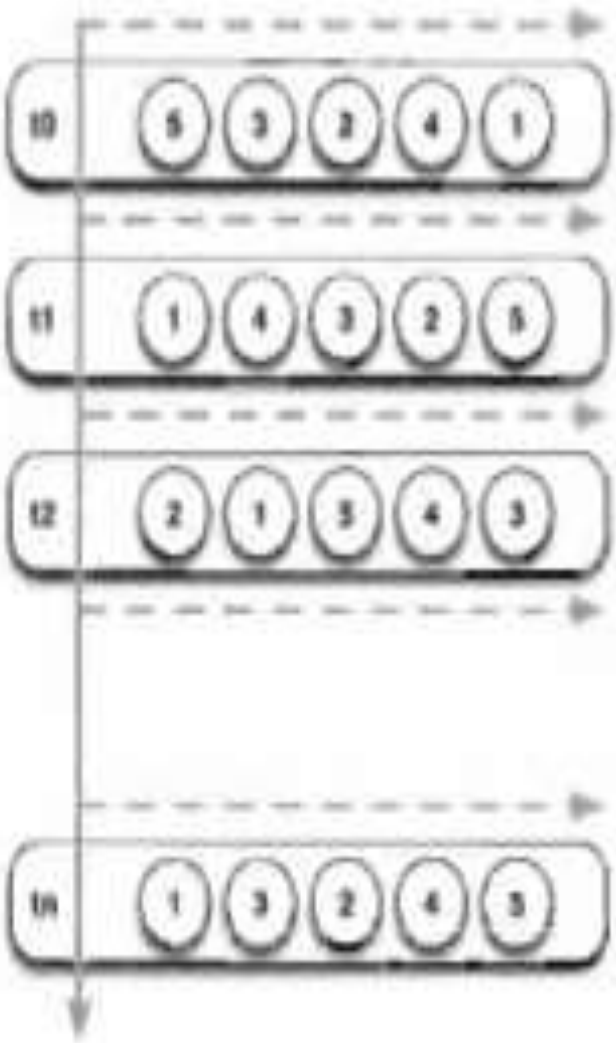
- Dans les modèles à base d' agents, à chacune des entités composant la simulation est associé un ensemble de comportements.
- Par exemple, dans dans MANTA, les comportements élémentaires associés aux agents fourmis incluent l'évaluation de leurs besoins, se déplacer, se nourrir, émettre et capter des phéromones, etc.
- Il est également possible de définir des comportements plus complexes, qui sont représentés par l'association de comportements élémentaires (par exemple la recherche de nourriture peut se faire par une séquence de perception des phéromones, de déplacement, puis d'émission d'autres phéromones).
- Du point de vue opérationnel, le modélisateur implémente ces comportements, en y associant les contraintes de temporalité de la simulation (choix temps discret/continu), comme un ensemble de changements d'états et/ou d'interactions entre entités dans le cas des comportements élémentaires, et d'une composition de comportements dans de cas de comportements plus complexes.
- Se pose alors un premier problème, celui de l'ordonnancement des agents, c'est-à dire le choix de l' ordre dans lequel ils sont activés.

Ordonnancement des agents





Stratégie naïve



Stratégie pseudo-parallèle

Ordonnancement des agents

- Soit donc une simulation contenant *n* agents. Une solution naïve d'implémentation, à supposer que l'ensemble des agents soit stocké dans un vecteur, est de parcourir la liste des agents et de les activer successivement
- L'algorithme est on ne peut plus simple (par exemple en Java, une boucle sur l'ensemble des pas de temps, qui appelle à son tour une boucle sur l'ensemble des agents) :

```
for (int i=O; i < tempsMaximum; i += pasDeTemps)
{for (Agent a- : listeAgents)
    {a.executer();}
}
```

Ordonnancement des agents

- Une autre solution est de choisir l'agent à exécuter selon un tirage aléatoire dans la liste, ou bien de mélanger la liste des agents à chaque pas de temps. Dans ce dernier cas, le code serait alors :

```
for (int i=O; i < tempsMaximum; i += pasDeTemps)
{for (Agent a - : listeAgents) {a.executer(); }
listeAgents.mélangerAléatoirement(); }
```

IMPLÉMENTATION

- **Programmation spécifique ou plate-forme générique?**
- Lors du démarrage d'un projet de modélisation à base d'agents, il peut être tentant de vouloir construire dans la foulée son propre simulateur

- Si le simulateur existe déjà, le modèle opérationnel sera bien une traduction du modèle conceptuel dans le « langage » (en terme de méta-modèle imposé) de la plate-forme.

- si chaque modèle est unique, ils partagent néanmoins tous (sauf cas particulier) de nombreuses composantes de leur métamodèle commun qui peuvent être décrites et implémentées de façon générique et réutilisées par chacun au prix d'un minimum d'efforts (et avec la certitude, de plus, que ces composantes auront été vérifiées dans d'autres réalisations, ce qui peut éliminer beaucoup de causes d'erreurs lors de l' implémentation).

IMPLÉMENTATION

L' utilisation d' une plate-forme existante quand cela est possible. Encore faut-il , bien entendu, qu'elle soit correctement choisie, ce qui n' est pas toujours évident à faire. Au-delà des besoins spécifiques que peut nécessiter un modèle, voici les caractéristiques à prendre en compte pour choisir une plate-forme :

- Existence d'un méta-modèle (ou d'un cadre abstrait) suffisamment documenté (de façon à pouvoir en hériter), et mise à disposition de bibliothèques d'objets réutilisables (comme les environnements, par exemple) s'appuyant sur ce méta-modèle.
- Mise à disposition d'une gamme suffisante d'outils «périphériques », comme les outils d'expérimentation et de visualisation mentionnés ci-dessus. La lecture et l'écriture dans des bases de données peuvent faire partie de ce type d'outils.
- Existence d'une communauté suffisamment nombreuse de modélisateurs (ou développement d'un nombre suffisamment important de modèles couvrant différents domaines scientifiques).
- Assurances que le développement de la plate-forme n'est pas le fait d'un seul individu, et qu'il existe une équipe, voire d'une institution ou d' une entreprise.

Panorama des principales plates-formes

➢ La liste des plates-formes de simulation à base d'agents disponibles à l'heure actuelle est à première vue très longue.

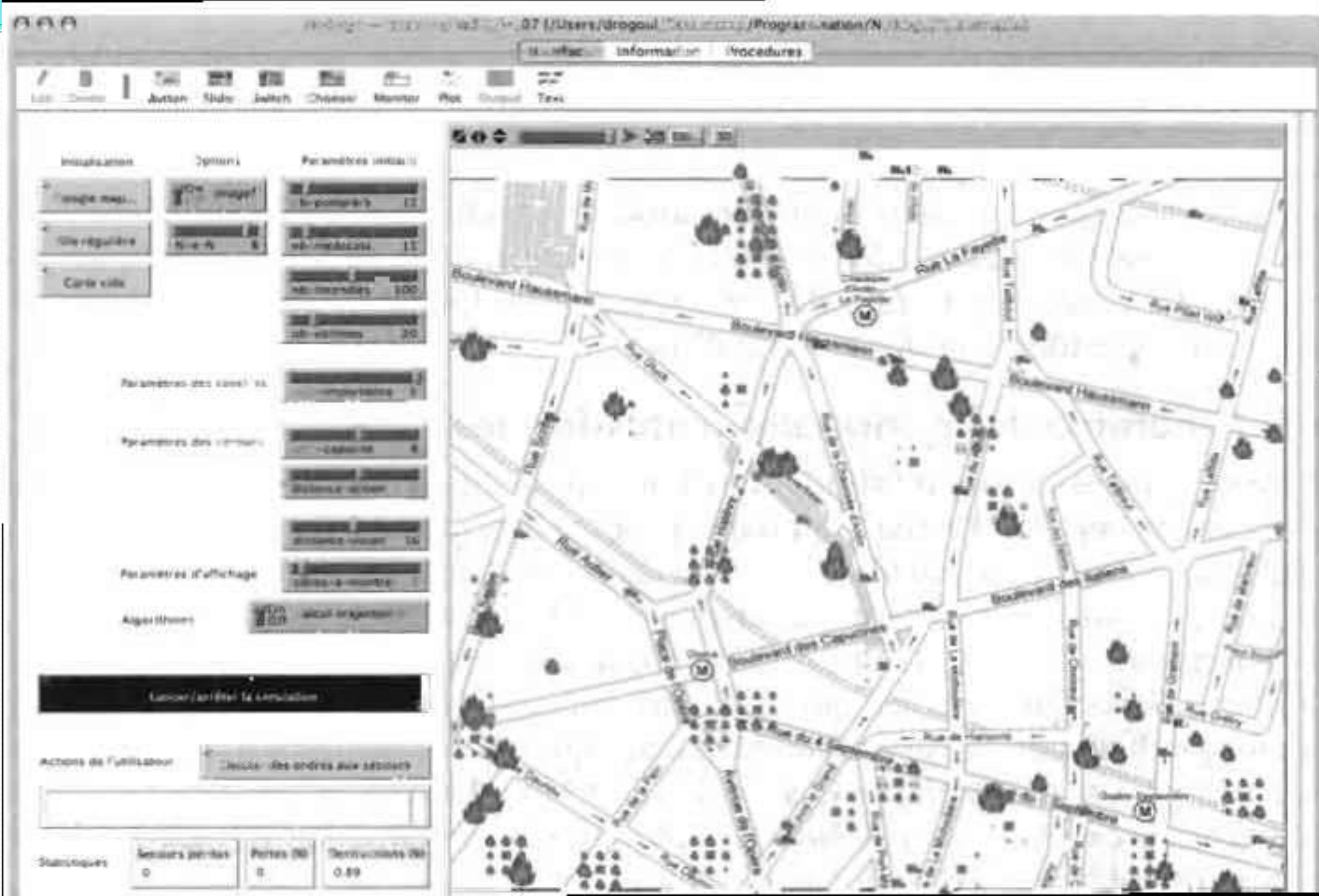
1. NetLogo

- La plate-forme NetLogo1, développée depuis environ dix ans (d'abord sous le nom de StarLogoT, puis depuis cinq ans en Java sous le nom de NetLogo), est l' héritière d'une longue tradition d'outils pédagogiques commencée avec Logo dans les années 1970 (pour l'apprentissage de la programmation et de l' algorithmique), puis StarLogo dans les années 1990 (apprentissage de la programmation parallèle, puis évolution vers les modèles à base d'agents).

- Elle inclut un langage de programmation dédié qui permet de manipuler des structures de haut niveau et des primitives qui réduisent considérablement l'effort de programmation demandé aux modélisateurs.

- NetLogo est particulièrement bien adaptée à un type de modèles en particulier :ceux composés d'agents mobiles qui évoluent en parallèle dans un environnement de type grille en deux dimensions, et dont le comportement est basé sur des interactions courtes. Mais elle n'est pas limitée à ce type de modèles et les dernières versions de la plate-forme permettent de créer des modèles en trois dimensions.

Panorama des principales plates-formes



Panorama des principales plates-formes

- NetLogo est structurée autour d' un méta-modèle très simple, qui comprend un ***Observer*** (l'équivalent d'une fusion du *Monde* et de l' *Environnement),* un ensemble de **Patch** (l 'équivalent des *Places)* et de ***Turtle*** (l ' équivalent des *Agents)* spécifiés par des ***Breed*** (l'équivalent d' *Espèces).*

- le langage de NetLogo n'est pas un langage orienté objet

- NetLogo est disponible pour tous les systèmes d'exploitation et les modèles s'exécutent de façon identique dessus.

Panorama des principales plates-formes

- NetLogo est structurée autour d' un méta-modèle très simple, qui comprend un ***Observer*** (l'équivalent d'une fusion du *Monde* et de l' *Environnement),* un ensemble de **Patch** (l 'équivalent des *Places)* et de ***Turtle*** (l ' équivalent des *Agents)* spécifiés par des ***Breed*** (l'équivalent d' *Espèces).*

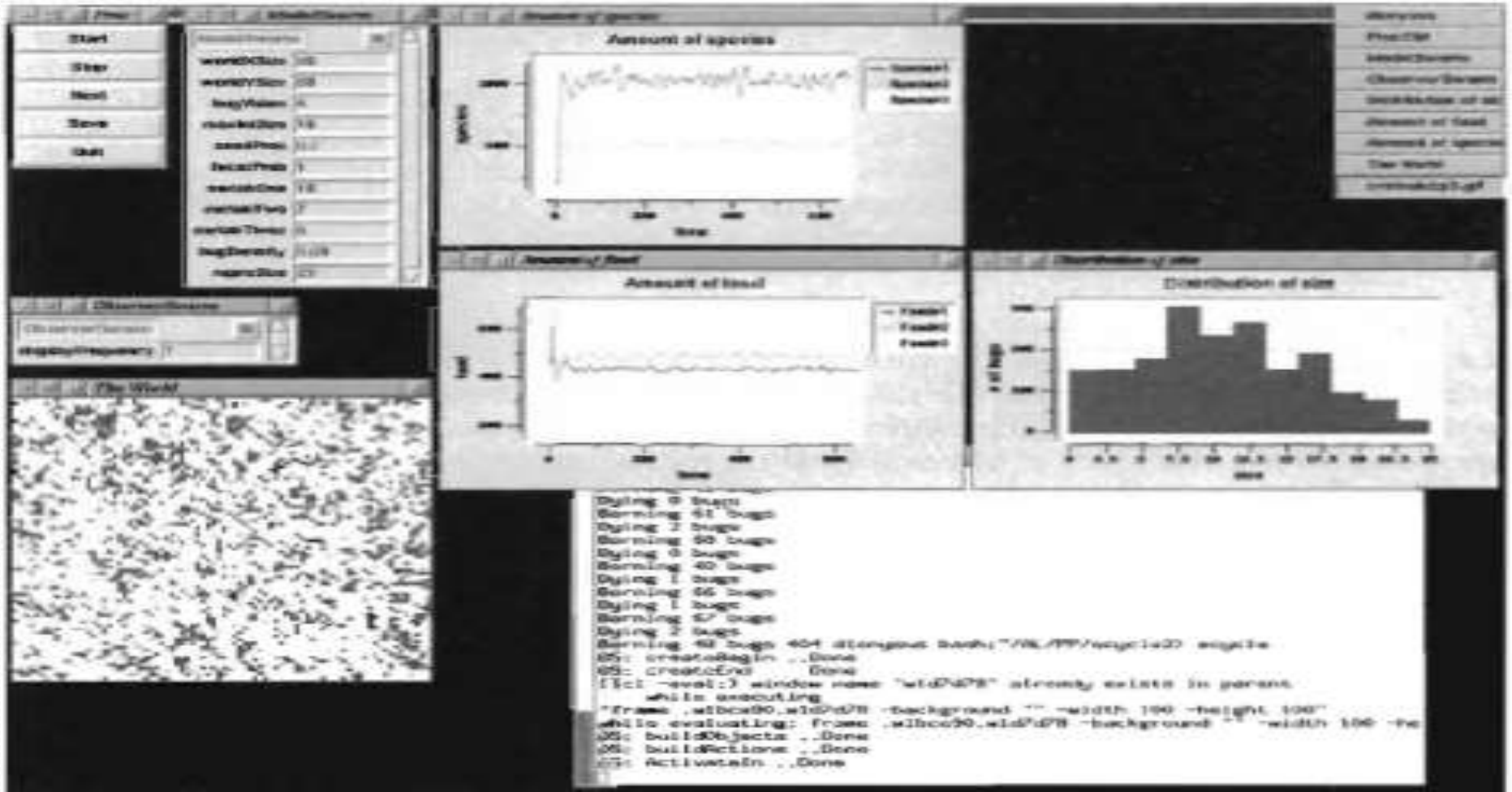
- le langage de NetLogo n'est pas un langage orienté objet

- NetLogo est disponible pour tous les systèmes d'exploitation et les modèles s'exécutent de façon identique dessus.

Panorama des principales plates-formes

Swarm

- Swarm2 , développée depuis le milieu des années 1990 en Objective-C (et, depuis peu, dans une version Java), s'est positionnée dès le début comme un environnement de développement et une boîte à outils générale pour les modèles à base d'agents.



Panorama des principales plates-formes

- Première plate-forme à avoir clairement séparé l'implémentation du modèle de celle d' un « laboratoire virtuel » permettant l'observation et la conduite d'expérimentations sur le modèle
- Un *swarm* est un groupe d'objets et un ensemble d'actions (doté de son propre ordonnancement) exécuté par ces objets.
- Tout *swarm* peut contenir d'autres *swarms,* et ceci jusqu 'au *swarm* principal constitué par l'observateur (l 'environnement d'expérimentation).
- Swarm, contrairement à NetLogo, ne se laisse cependant pas maîtriser facilement,surtout par des modélisateurs non spécialisés en informatique
- Dans un contexte informatique clairement dominé aujourd'hui par les langages Java et C++, Swarm nécessite de passer (pour des modèles complexes) à la programmation en Objective-C,

Panorama des principales plates-formes

Repast

- Le développement de Repast3, initié au début des années 2000, a été motivé par plusieurs objectifs :
- le premier était le développement d' un outil dédié aux modèles à base d'agents
- Le second était de procurer un environnement de simulation adéquat pour les sciences humaines (depuis la géographie humaine jusqu'à la sociologie ou l'économie) en proposant outils et abstractions spécialisés pour ces domaines
- Le troisième, enfin, était de faciliter l'accès à la modélisation pour les non-informaticiens en proposant un squelette de modèle facilement réutilisable et la programmation d'agents par l'intermédiaire du langage Python, plus simple à maîtriser que Java.
- Repast aujourd'hui est une impressionnante boîte à outils plutôt qu 'une plate-forme. Elle n'impose aucun modèle particulier, mais propose des facilités dont aucune autre plate-forme ne di spose, comme par exemple:

- le couplage avec des systèmes d'information géographiques (via la bibliothèque *OpenMap),*
- la possibilité de générer des statistiques en temps réel (via le langage *R),*
- la possibilité de gérer des modèles comprenant plusieurs types d'environnement simultanément
- la possibilité de contrôler et de paramétrer des simulations par lot ou encore celle de spécifier très précisément le type d'ordonnancement souhaité pour les agents.

Panorama des principales plates-formes

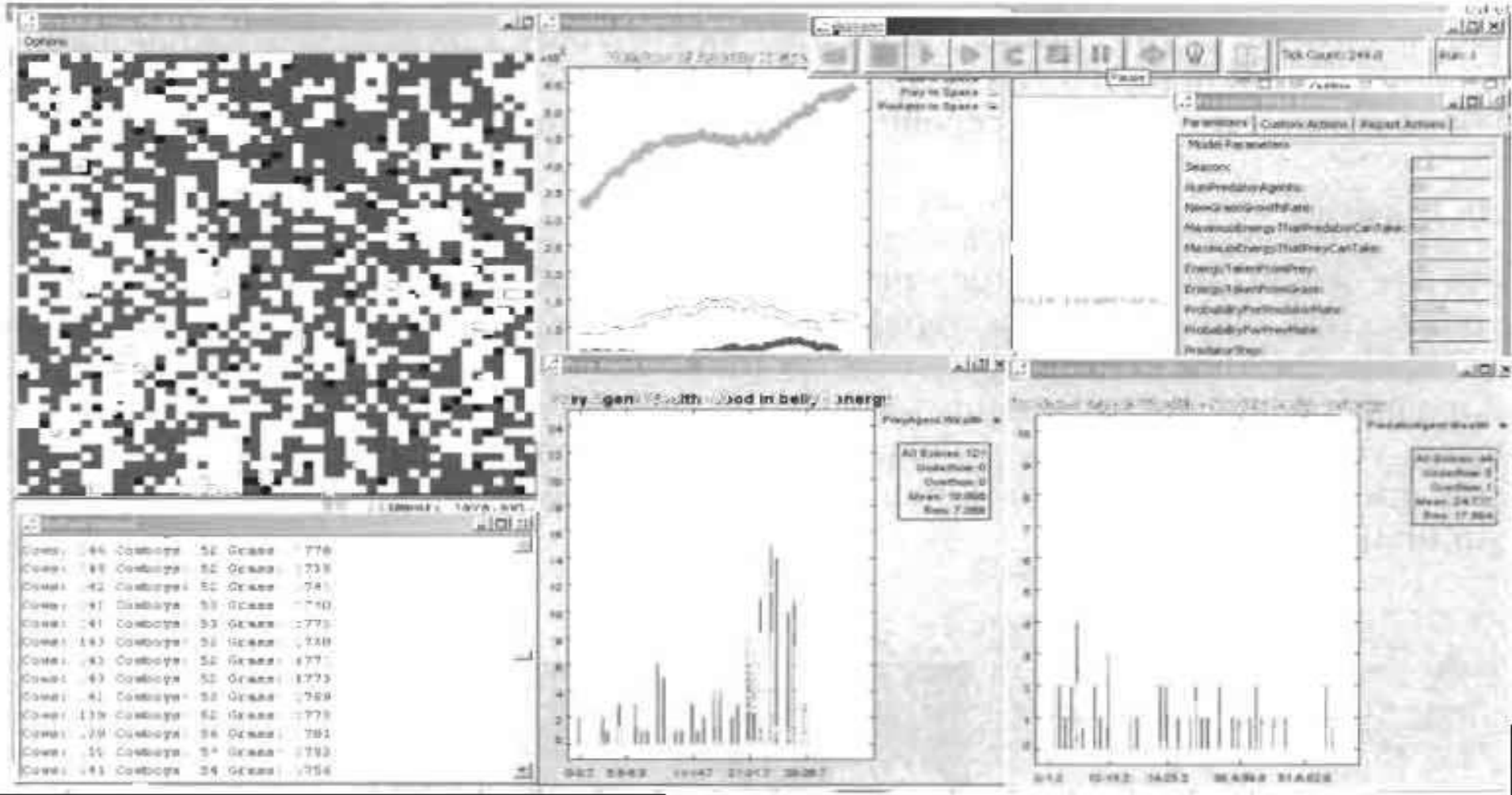
Repast

- Le développement de Repast3, initié au début des années 2000, a été motivé par plusieurs objectifs :
- le premier était le développement d' un outil dédié aux modèles à base d'agents
- Le second était de procurer un environnement de simulation adéquat pour les sciences humaines (depuis la géographie humaine jusqu'à la sociologie ou l'économie) en proposant outils et abstractions spécialisés pour ces domaines
- Le troisième, enfin, était de faciliter l'accès à la modélisation pour les non-informaticiens en proposant un squelette de modèle facilement réutilisable et la programmation d'agents par l'intermédiaire du langage Python, plus simple à maîtriser que Java.
- Repast aujourd'hui est une impressionnante boîte à outils plutôt qu 'une plate-forme. Elle n'impose aucun modèle particulier, mais propose des facilités dont aucune autre plate-forme ne di spose, comme par exemple:

- le couplage avec des systèmes d'information géographiques (via la bibliothèque *OpenMap),*
- la possibilité de générer des statistiques en temps réel (via le langage *R),*
- la possibilité de gérer des modèles comprenant plusieurs types d'environnement simultanément
- la possibilité de contrôler et de paramétrer des simulations par lot ou encore celle de spécifier très précisément le type d'ordonnancement souhaité pour les agents.

Panorama des principales plates-formes

- Il est à noter qu'une nouvelle version de Repast, baptisée Repast Simphony, a été dévoilée en décembre 2007. Elle inclut de nombreuses extensions, en particulier du point de vue de la description de l'environnement et de sa topologie.
- Une bonne maîtrise de Java reste cependant obligatoire pour tirer parti de toute sa richesse.

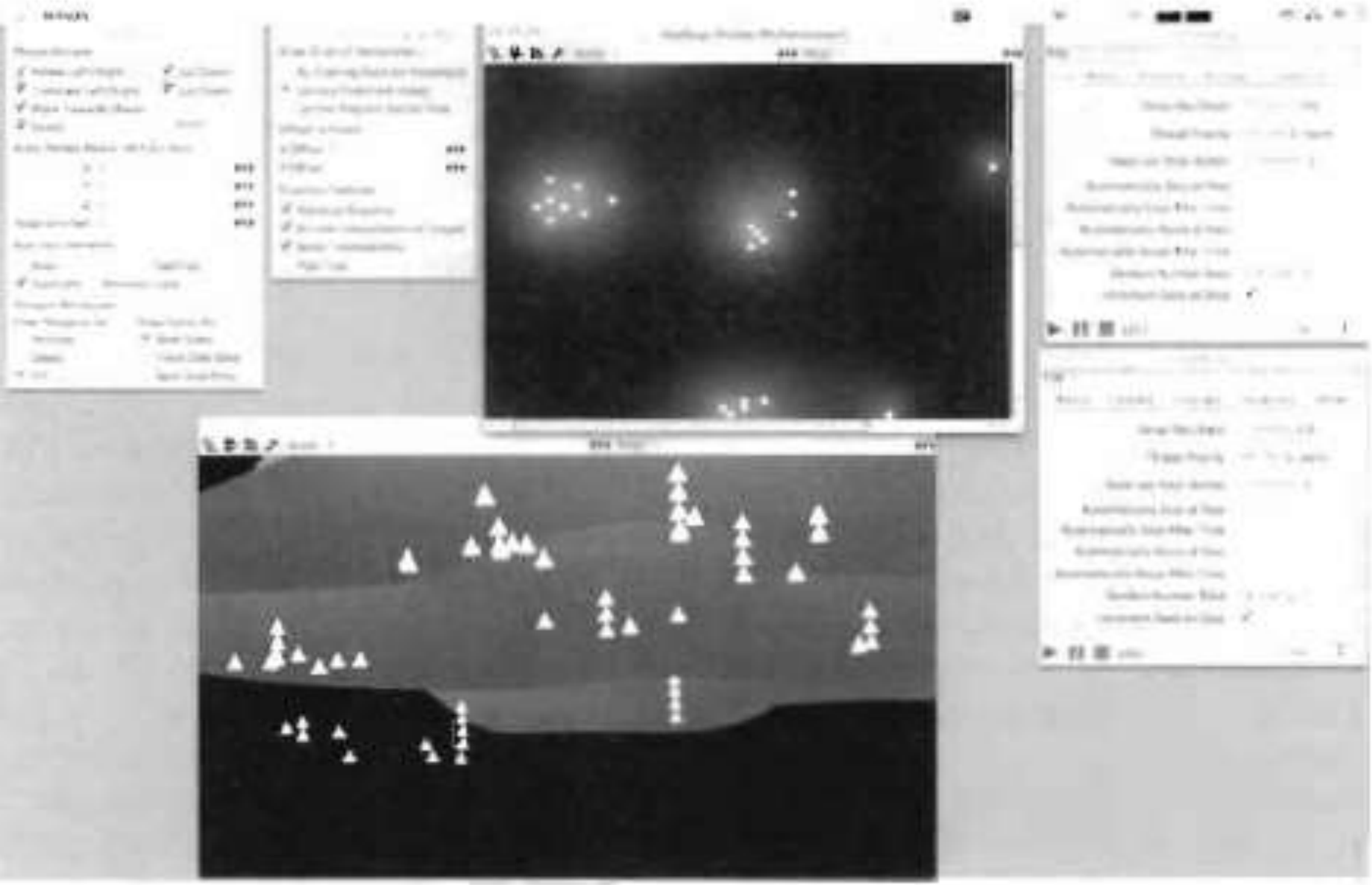


Panorama des principales plates-formes

Mason

- Mason5 est la plus récente des plates-formes évoquées ici. Elle a été développée comme une alternative plus rapide (et plus légère) à Repast, dont elle reprend quelques caractéristiques (en termes de modélisation de l'environnement ou d'ordonnancement).le premier était le développement d' un outil dédié aux modèles à base d'agents
- Ses trois points forts sont sa rapidité d'exécution, son indépendance totale vis-à-vis des systèmes d'exploitation et sa capacité à gérer des modèles physiquement réalistes en trois dimensions (par !'inclusion de moteurs physiques de simulation).
- Mason n'arrive cependant pas à égaler ce qu'offrent Swarm ou Repast, et la communauté de modélisateurs qui l'utilisent semble réduite.
- Mais il peut être intéressant de l' utiliser pour des modèles très spécifiques, par exemple ceux comportant un très grand nombre d'agents ou imposant une description réaliste d'un environnement physique.
- Une autre faiblesse réside dans ce que Mason n'inclut que peu d'outils d'analyse et de suivi comme ceux développés pour Repast,

Panorama des principales plates-formes



Panorama des principales plates-formes

Cormas

- Cormas6 est une plate-forme développée par le CIRAD au milieu des années 1990, avec pour ambition de modéliser et de simuler les rapports entre acteurs et ressources naturelles renouvelables.
- Elle dispose d'outils d' inspection, de liaison avec les bases de données, de visualisation et de programmation qui en font un choix très intéressant.
- Cormas propose de base un méta-modèle qui n'est pas sans rappeler celui de NetLogo, et qui permet, en très peu de temps, de construire des modèles de complexité moyenne.
- une des grandes forces de Cormas est de s'appuyer sur le langage Smalltalk, qui reste inégalé en termes de facilité de programmation et de conception, et qui permet surtout de réutiliser extrêmement facilement les nombreuses extensions déjà développées pour d'autres modèles