Un langage à base de logique floue pour la simulation de comportements individuels d'animaux

A fuzzy logic based language to simulate animal's individual behaviors

Ludovic Coquelle Cédric Buche Pierre Chevaillier
Centre Européen de Réalité Virtuelle (CERV)
EA2215 UBO-ENIB

CERV, 25 rue Claude Chappe - BP38 F - 29280 Plouzané France
{coquelle, chevaillier, buche}@enib.fr

Résumé :

behavioRis est un environnement de modélisation individu-centrée de comportement animal. Le but est de traduire simplement les descriptions éthologiques de comportement en simulations multi-agents. Pour cela, d'une part une architecture modulaire d'agents réactifs permet de simuler les comportements des *animats* (modèles d'animaux) en utilisant la notion de *processus inné de déclenchememnt* ; les composants de cette architecture définissent séparément les perceptions, les comportements, le contrôle global, et une représentation de l'état interne sous forme de FCM (Fuzzy Cognitive Map) simple à manipuler pour un expert du domaine. D'autre part, un langage spécifique basé sur la logique floue permet de décrire les modèles et d'exprimer les comportements. Une application à la modélisation de comportement d'évitement de poisson face à un engin de pêche illustre son utilisation.

Mots-clés :

Comportement animal, langage, logique floue, modélisation individu-centrée, simulation, agent réactif.

Abstract:

behavioRis is an environment for simulation of animal's behaviours using individual-based models. This simulator is based on multi-agents system technology and integrates *animats* (animal's representations) as reactive autonomous agents. Its goal is to use innate releasing mechanism as triggers of behaviour's manifestations considering ethological descriptions of animal's behaviours. It proposes an architecture designed in modular components : the first describes perception's possibilities of the *animat*, the second gathers each of its behaviours, the third controls synthesis of movement, and the last is a representation of its internal state. On top of that a language based on fuzzy logic permits an easy expression of behaviours. To illustrate our model we present an application showing simulation of fish escape behaviours.

Keywords:

Animal's behaviour, language, fuzzy logic, individual-based modelling, simulation, reactive agent.

Introduction

L'éthologie étudie le comportement naturel de l'animal dans son milieu en se focalisant au niveau de l'individu [22, 15]. Des descriptions d'éthologues résultent des modèles de comportement individu-centrés. Pour simuler ces modèles, l'utilisation des systèmes multi-agents est une technique informatique privilégiée. Plus spécifiquement, les architectures réactives d'agents autonomes, qui s'appuient sur des règles simples de type *cause–conséquence*, peuvent être utilisées pour simuler les modèles éthologiques, qui eux s'appuient sur la notion de réaction instinctive ou plus précisément sur la notion de processus inné de déclenchement [15].

Différentes solutions d'architectures logicielles ont été développées pour ces systèmes de type réactif, que ce soit en robotique [6, 1, 3] ou en animation [21, 2, 13]. Cependant le lien entre ces architectures et les descriptions éthologiques de comportements animaux n'est pas explicite. C'est dans ce but qu'à été développé *behavioRis*. A la fois architecture d'agents réactifs et langage, *behavioRis* propose un outil de description de comportements innés d'animaux. Ce langage permet de modéliser (exprimer puis simuler) un comportement animal en se basant sur les concepts utilisés en éthologie, à savoir la description des perceptions, des stimuli externes et internes, et des réponses motrices stéréotypées de comportement qui corres-

pondent aux *fixed action patterns* [10].
Les connaissances en éthologie sont essentiellement qualitatives et reposent sur des données imprécises (par exemple les notions de distance inter-individuelle, de niveau de fatigue). Avec comme objectif un langage de modélisation pour la simulation, l'utilisation d'un système d'inférence floue permet de traduire ces données ainsi que les règles de causalité des modèles éthologiques. De plus, les connaissances étant parcellaires, le système d'inférence floue permet de généraliser le modèle pour le simuler. Le langage *behavioRis* utilise la logique floue pour fournir un moyen de spécifier complètement un comportement au niveau de chaque individu et de l'exécuter.
La première partie de cet article présente quelques aspects des simulations de comportement d'entités réactives et fixe les objectifs. Dans une deuxième partie, l'architecture d'agent est détaillée de façon à introduire la présentation du langage dans la partie suivante. Enfin, une dernière partie porte sur les applications actuelles de modélisation des réactions de poissons face à un engin de pêche (vu comme un danger par le poisson) en étudiant particulièrement les trajectoires d'évitement.

1 Simulations de comportement d'animaux

Le modèle des BOIDS[19] de REYNOLDS est un des modèles individu-centrés les plus connus dans ce domaine. En combinant trois comportements individuels en réaction à la perception des quelques plus proches voisins (attraction, répulsion, et orientation), le modèle restitue un comportement de déplacement en groupe. Pour ce type de modélisation, la simulation informatique s'inscrit dans l'approche biologique d'expérimentation des modèles [16, 11]. En effet, la simulation peut avoir comme but d'être prédictive, comme l'outil CORMAS[5], environnement de développement de système multi-agents pour les problèmes de dynamiques et d'usage des ressources. Mais elle peut aussi avoir un but explicatif en permettant de tester des modèles comportementaux, ou simplement des hypothèses particulières de comportement. MOBIDYC[12] est un exemple d'environnement de modélisation de comportement animal qui permet de construire, sous forme d'automate cellulaire, une simulation 2D du comportement d'entités autonomes. Plus généralement, les outils utilisables pour ce type de modélisation font partie de l'étude de *la vie artificielle*, dont WANG donne un aperçu dans [24].

behavioRis, qui est présenté dans la suite de cet article, est un environnement pour développer des modèles de comportement individu-centrés. Il est orienté simulation, et pour cela propose une représentation 3D du monde. Son objectif n'est pas de simuler un modèle physiologique d'animal, mais plutôt des règles comportementales issues des descriptions éthologiques qui s'appuient au maximum sur les données observables. L'échelle de modélisation visée est celle de l'individu, avec comme objectif principal la modélisation du déplacement. Pour cela, il reprend les notions de stimuli, interne ou externe, et d'action stéréotypée pour exprimer des réponses comportementales liées à un état perceptif précis. Cette orientation réactive de la simulation permet de simuler dans un univers 3D complexe le comportement d'agrégats d'animaux.

Les entités réactives de *behavioRis* représentent des modèles réactifs d'animaux que l'on veut simuler. Cela inclut :
– une architecture d'agent qui est le support à l'assemblage des différents composants du modèle d'animal pour le rendre exécutable ;
– un langage qui permet de décrire ce modèle, et particulièrement les comportements.

2 L'architecture *behavioRis*

On trouve de nombreux exemples d'architectures d'agents autonomes réactifs, que ce soit dans le domaine de la robotique [6, 1], ou dans d'autres domaines tel celui de la réalité virtuelle [2, 13]. La logique floue a été utilisée dans ces domaines aussi bien pour définir des compor-

tements que pour définir des règles d'activation de ces différents comportements souvent concurrents et s'exécutant dans un environnement dynamique [20, 9, 3].

behavioRis propose une architecture servant de base à un langage reprenant les concepts des modèles éthologiques. Dans *behavioRis*, un modèle d'animal est représenté par l'objet nommé *animat* (terme introduit par WILSON [25]). Un *animat* est un objet actif situé dans l'espace, dont le comportement global est défini par quatre types de composants. Cette décomposition componentielle permet de décrire un modèle d'animal en utilisant des éléments modulaires réutilisables. La figure 1 représente notre architecture globale d'*animat* en interaction avec son environnement. A tout instant de la vie d'un *animat*, tous ces composants sont activés dans un ordre aléatoire. Chaque type de composant est maintenant brièvement décrit.

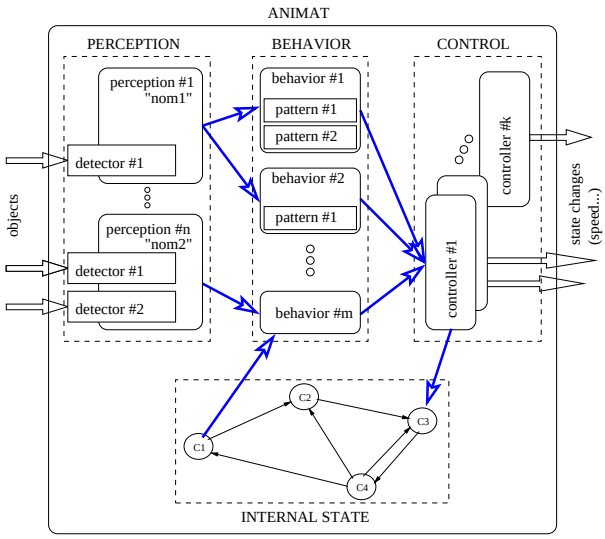

Figure 1 – Architecture *behavioRis* d'un *animat*

Perception

Les composants *perception* modélisent les stimuli externes. Ils reçoivent leurs données de l'environnement et en sortie associent des valeurs numériques à chaque objet perçu. Ils sont paramétrés avec un nom qui a pour rôle de leur associer une sémantique dans le modèle comportemental. Par exemple, on pourra créer une perception nommée "danger". La définition de leur capacité à percevoir se fait en leur associant des détecteurs prédéfinis, eux-mêmes paramétrés par un moyen de détection (par exemple l'utilisation d'un graphe de visibilité) et des valeurs limites (domaine de perception).

Etat interne

Le composant unique *internal state* modélise les stimuli internes. Un stimulus interne est ici tout état endogène de l'individu comme les excitations sensorielles internes, les hormones, les rythmes endogènes ou chronobiologiques... L'état interne est donc un ensemble d'états, nommés comme pour les perceptions, chaque état prenant une valeur numérique. Ils peuvent être structurés sous forme de graphe d'influence en utilisant le principe des FCM (Fuzzy Cognitive Map [14]). Les FCM ne fournissent pas ici une représentation spatiale du monde mais sont utilisées de la même manière que [7] qui propose un modèle de contrôle de comportement de poissons. Une FCM permet de décrire l'évolution des états émotionnels d'un *animat* [17] avec l'avantage de rester déclaratif, explicatif, et de pouvoir être spécifiée par un non-spécialiste en informatique. Les états internes de cette FCM sont utilisés par les éléments de comportement comme des stimuli internes. Les relations entre concepts deviennent alors des relations d'influence (activation, inhibition) entre les éléments de comportement.

Comportement

Les composants *behaviors* modélisent les réponses comportementales innées et stéréotypées. Ce sont des objets actifs, qui accèdent aux perceptions et aux états internes.
En fonction de ces entrées, un composant *behavior* évalue une réponse composée d'une commande positionnelle (vitesse et orientation), d'une commande de modification de l'état interne, et d'un degré d'activation de ce *behavior*. L'*animat* est dirigé suivant le modèle physique *simple vehicle* utilisé par REYNOLDS [18].

Les comportements sont indépendants mais peuvent utiliser les valeurs de l'état interne. Cet état interne n'est pas strictement un contrôleur des comportements mais ses règles d'influence permettent de spécifier des relations entre variables d'état et donc d'assurer une cohérence dans les comportements.
Le composant *behavior* décrit ici de manière générique est, dans nos simulations, implémenté par un contrôleur flou. Le fonctionnement du comportement flou est détaillé dans la partie traitant du langage. Notons cependant la notion de *pattern* qui permet de nommer un sous-comportement, que nous assimilerons plus tard à un ensemble de règles floues.

Contrôleur

Enfin, le dernier type de composant, les *controllers*, synthétise les réponses des comportements pour les transformer en actions. On peut utiliser différents types de contrôleur, que ce soit pour faire de la fusion ou de la sélection d'action (par exemple en fonction du degré d'activation), avant ou après la défuzzyfication des variables linguistiques de sortie des comportements. Les *controllers* ont entre autres pour rôles de calculer la nouvelle position de l'*animat* et de modifier l'état interne en conséquence de la réalisation des actions.

3 Un langage déclaratif de comportement animal

Un langage peut être une interface de description du comportement. CML [8] est un exemple de langage pour décrire un comportement en terme de successions et dépendances d'actions, avec un moteur d'inférence pour la planification de ces comportements. MOBIDYC [12] propose le langage ATOLL pour décrire une modélisation, dans laquelle les comportements sont décrits par la succession des tâches prédéfinies.

L'architecture *behavioRis* présente un langage adapté à la description de modèles éthologiques d'animaux. Ce langage utilise la logique floue pour transcrire la modélisation des comportements en se basant sur les composants de l'architecture. A ce niveau, il faut différencier la modélisation, qui est associée à l'*espèce*, de la simulation, associée à l'*animat*, comme illustré figure 2. Si l'*animat* et l'*espèce* ont une architecture totalement similaire, l'objectif étant de simuler un nombre important d'individus (de l'ordre de plusieurs centaines), le langage manipule le modèle (et donc la notion d'*espèce*), alors que l'environnement a pour rôle de créer le nombre souhaité d'individus (c'est-à-dire d'*animats*) exécutant ce modèle. Une des caractéristiques du langage est de permettre l'assimilation entre *espèce* et *animat*. Par exemple, au niveau des *comportements flous*, si une règle manipule la notion de *perception* dans la modélisation c'est-à-dire une *perception* de l'*espèce*, cette règle doit dans l'exécution de l'*animat* manipuler la *perception* réelle de l'individu. Ainsi, le langage permet facilement de décrire les comportements de tous les individus d'une même espèce en intégrant les facilités d'expression de la logique floue.

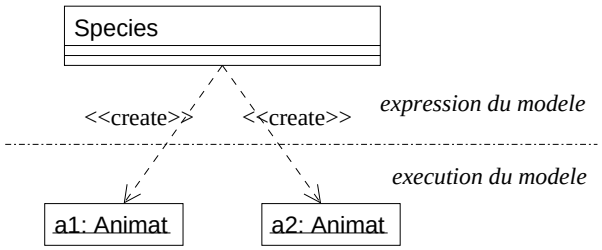

Figure 2 – Séparation entre le modèle et sa simulation

Description générale du modèle

La description d'une espèce a pour rôle de déclarer les éléments de l'architecture. Le code 1 en donne un exemple simple. Cet exemple définit l'espèce "SpStrangeAnimat" dotée de deux propriétés, un état interne sous forme de FCM, une perception nommée "danger", un comportement nommé "repulsion", et un contrôleur de déplacement. Tous ces éléments sont définis dans ce code de déclaration de l'espèce, excepté le comportement définit ci-après.

Code 1 – Exemple de définition d'une espèce.

```
Species SpStrangeAnimat :
  Property length = gaussRand(0.12, 0.05)
  Property fearDist = "down_ramp(5, 12)"
  InternalState Fcm() :
    Concept idle
    Concept afraid
   afraid influence idle with_factor -0.80
  End
  Perception danger :
    Rate 0.2
    Detector Simple(dist=120)
  End
  Behavior repulsion
  Controller BasicMovement() :
    Rate 0.1
  End
End
```

Le langage de comportement

La définition d'un élément de comportement est moins directe. C'est à ce niveau que la logique floue va servir à transcrire les règles de comportement issues des descriptions littérales de l'éthologie. Le but est de décrire un comportement instantané causé par la présence de stimuli, cette réaction se traduisant par une action sur l'état interne et une action sur les propriétés de l'*animat*.

Chaque comportement de type flou est en fait une base de règles floues au niveau de l'espèce et sera instancié en système d'inférence floue réel au niveau de l'*animat*. Le code 2 donne un exemple de définition d'un comportement flou. Ce composant *behavior* est celui qui est référencé dans le code 1 par le nom "repulsion".

C'est un composant actif de l'architecture. "Rate" définit sa période d'activation en seconde (temps entre deux cycles de calcul d'inférence) pendant la simulation.

La configuration du contrôleur, qui n'apparait pas ici car elle est paramétrée par défaut, permet de spécifier les types de défuzzyfication (centre de gravité par défaut) et les opérateurs d'implication (Larsen par défaut), de combinaison de règles (somme bornée par défaut), de conjonction (produit par défaut), et de disjonction (maximum par défaut), suivant les principes de la logique floue [4].

La deuxième partie, "Linguistic", définit des valeurs linguistiques floues. La notation standard FCL (Fuzzy Control Language) n'a pas été reprise pour garder l'homogénéïté avec le reste du langage. On retrouve cependant les utilisations habituelles d'ensembles flous définis par des fonctions d'appartenance linéaires sur intervalles contigüs, avec des raccourcis d'écriture pour les profils classiques comme les triangles, les trapèzes, les rampes *etc.* L'uniformité du langage a aussi pour rôle de garder une syntaxe simple, surtout pour l'expression des règles, où l'idéal serait de reprendre textuellement les descriptions éthologiques.

Les entrées du contrôleur sont désignées par la partie "stimuli". Ce sont les perceptions et les valeurs d'état interne, qui sont fuzzyfiées en singleton pour créer les variables floues utilisables en tant que valeurs dans les règles. Ainsi, le premier stimuli définit la variable "fear" comme la fuzzyfication de l'état interne "afraid". Au niveau de l'*espèce*, cette association est uniquement symbolique, mais deviendra un réel senseur au niveau de l'*animat*. Il en est de même pour les perceptions, excepté que les perceptions peuvent renseigner le type d'objet perçu ainsi que sa position relative (coordonnées sphériques de l'objet perçu dans le repère de l'animat qui possède le comportement) et sa vitesse instantanée. La solution adoptée est de créer dynamiquement (si besoin) des variables floues correspondant à chaque donnée que peut fournir une perception. Dans l'exemple, "enemy" est une variable liée à la perception "danger", ce qui définit (potentiellement) les variables floues "enemy.distance", "enemy.yaw"...

Ce comportement inclut deux ensembles de règles, nommés "detection" et "avoidance".

Les règles de comportement

Les règles floues sont définies dans les *behavior patterns*. Le rôle de ces *patterns* est de regroupper des règles ayant ensemble un sens comportemental, en vue de pouvoir réutiliser ce *pattern* dans divers comportements. Des exemples de *patterns* sont donnés dans les codes 3 et 4.

Les sorties du contrôleur (actuateurs) sont pré-

Code 2 – Exemple de définition d'un comportement flou.

```
Behavior FuzzyRules() repulsion :
  Rate 0.1
  Linguistic :
    smallDist   is down_ramp(5,12)
    speedAvoid  is triangle(1.7,1.9,2.1)
    free        is up_ramp(0.2, 0.6)
    worried     is triangle(0.5,1.0,1.2)
  End
  Stimuli :
    fear  is internalState("afraid")
    idle  is internalState("idle")
    enemy  is perception("danger")
       undef(enemy.distance)  is 10000
  End
  BehaviorPattern detection
  BehaviorPattern avoidance
End
```

définies par l'architecture. Ce sont les valeurs du degré d'activation (variable "activation"), de la commande en vitesse (variable "speed" avec ses trois sélecteurs "x", "y" et "z") et orientation (variable "orientation" avec les sélecteurs "roll", "pitch", "yaw"), et des effets sur l'état interne (variable "state" avec pour sélecteur le nom de l'état interne).
Il est également possible d'utiliser des modificateurs de valeur floue (fonctions floues), comme "oppositeAngle".

Code 3 – Exemple de définition d'un pattern de comportement à deux règles.

```
BehaviorPattern detection :
  if idle  is free
  then activation  is null
  end
  if enemy.distance  is smallDist
  then state.afraid  is worried
  end
End
```

4 Application

behavioRis est utilisé actuellement pour le calcul de trajectoires d'évitement de poissons face à un chalut de pêche ou à un robot sous-marin (ROV pour Remote Operating Vehicle).
Des campagnes en mer ont pour but d'estimer l'abondance des animaux marins par observation directe [23]. Lors de ces observations, des

Code 4 – Exemple de définition d'un pattern de comportement.

```
BehaviorPattern avoidance :
  if fear  is worried
  then
    speed.x  is speedAvoid
    and orientation.yaw  is
        oppositeAngle(enemy.yaw)
    and activation  is one
  end
End
```

comportements en réaction à la présence du robot sous-marin ont été observés. Nous cherchons à reproduire ces comportements. La finalité de cette simulation est de pouvoir corriger les estimations d'abondances avec les informations de la simulation pour les individus non-observables.

Pour cette simulation, nous avons mis en place différents patterns de comportements pour trois espèces de poisson, et simuler les déplacements du ROV. Le modèle des comportements a été exprimé par un expert de manière littérale. Par exemple, pour l'espèce *Coryphaenoides rupestris* (Grenadier), le comportement décrit à l'approche du ROV est le suivant : *«Passe en état d'alerte sans nager beaucoup mais avec des petits mouvements brutaux, puis s'enfuit plutôt vers le bas à 30^o et vers l'avant, et garde une distance de sécurité (le* ROV *peut le suivre car le comportement perdure au delà de 10m)»*. Le comportement complet de cette espèce est traduit dans *behavioRis* avec un *animat* à deux états internes qui sont "idle" et "afraid", une perception "danger" détectant tout objet mobile dans un périmètre défini, et un comportement "DangerReaction" composé de quatre *BehaviorPatterns* que sont "NoReaction", "ROVDetection", "DownAvoidance", et "LateralOppositeAvoidance". Le code 5 représente la traduction dans le langage *behavioRis* de la fuite avec un angle de 30^o.
La dynamique de la FCM d'états internes et l'utilisation du flou pour décrire les valeurs seuils et les règles de causalité ont permis de reproduire un comportement crédible du point de

Code 5 – Définition d'un pattern de comportement d'évitement vers le bas.

```
BehaviorPattern DownAvoidance :
  if fear is reallyAfraid
  then
    speed.x is speedAvoid
     and orientation.pitch is
       pitchGlobal(angleAvoid)
     and activation is fear
  end
End
```

vu de l'expert. La figure 4 reproduit également un comportement émergent de l'espèce *Squalid sharks*. Ce comportement de fuite, obtenu au cours d'une simulation dont la figure 3 donne un aperçu, existe dans la réalité et est connu sous le nom *flash expansion*.

Figure 3 – Capture d'écran de la simulation du comportement *flash expansion* de l'espèce *Squalid sharks*

Conclusion

Les éthologues utilisent le schéma causal pour décrire de manière externe les causes endogènes

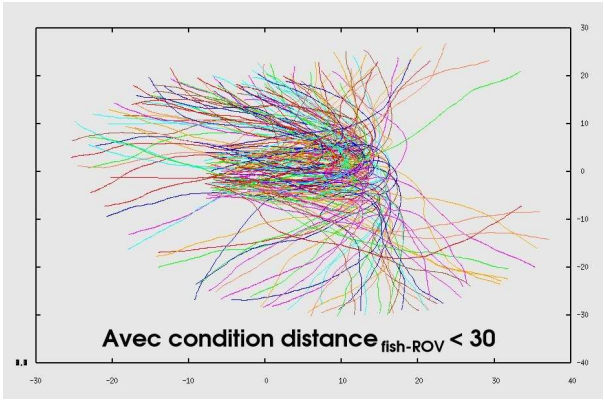

Figure 4 – Flash expansion : Chaque courbe symbolise la trajectoire d'un poisson ; au départ tous en banc, les poissons se dispersent rapidement à l'approche du ROV (se déplaçant ici de la droite vers la gauche).

ou exogènes des comportements. A partir de ces descriptions des réactions instinctives (leurs causes et leurs manifestations) au niveau d'un individu, les modélisations centrées individu permettent de simuler des systèmes comportementaux complexes dans des groupes d'individus.

behavioRis est un environnement réactif pour implémenter ces modèles individu-centrés. Sa modularité pour les perceptions et les comportements permet de construire des modèles à partir des descriptions des éthologues, et son module de gestion d'état interne par FCM traduit les modèles internes de causes/conséquences de manière simple pour les experts du domaine. Avec l'utilisation de la logique floue, le langage offre au modélisateur la possibilité d'exprimer un comportement à partir de données imprécises tout en maîtrisant le processus de contrôle résultant.

Les applications de modélisation de poissons donnent un exemple d'utilisation pour l'analyse de comportements d'évitement.

Il reste encore à expérimenter dans *behavioRis* la mise en place d'éléments de comportement particuliers comme les décisions associées à la limite de plusieurs domaines d'application de comportement ou les effets de cycle entre com-

portements. L'aspect déclaratif du langage permettrait également de mettre en place une interface utilisateur pour visualiser le modèle (l'aspect composant et FCM s'y prêtent naturellement) et même d'assister sa création.

Remerciements :

This study has been carried out with the financial support from *Région Bretagne* and from the Commission of the European Communities, Agriculture and Fisheries (FAIR) specific RTD programme, CT96 1555, £Development of predictive model of cod-end selectivity£. It does not necessarily reflect its views and in no way anticipates the Commission£s future policy in this area.

Références

[1] Ronald C. Arkin, *Motor schema based navigation for a mobile robot : An approach to programming by behavior*, IEEE Conference Robotics and Automatics (Raleigh,NC), april 1987, pp. 264–271.

[2] Bruce Mitchell Blumberg, *Old tricks, new dogs : Ethology and interactive*, Ph.D. thesis, Massachussets Institute of Technology, 1997.

[3] Andrea Bonarini, Giovanni Invernizzi, Thomas Halva Labella, and Matteo Matteucci, *An architecture to coordinate fuzzy behaviors to control an autonomous robot*, Fuzzy Set and System, vol. 134, 2003, pp. 101–115.

[4] Bernadette Bouchon-Meunier and Christophe Marsala, *Logique floue, principes, aide à la décision*, Informatique et systèmes d'information, Hermes, 2003.

[5] François Bousquet, Innocent Bakam, Hubert Proton, and Christophe Le Page, CORMAS *: Common-pool resources and multi-agent systems*, Lecture notes in Computer Science **1416** (1998), 826–837.

[6] Rodney A. Brooks, *A robust layered control system for a mobile robot*, IEEE Journal of Robotics and Automation **2** (1986), no. 1, 14–23.

[7] J. A. Dickerson and B. Kosko, *Virtual worlds as fuzzy cognitive maps*, Presence **3** (1994), no. 2, 173–189.

[8] John Funge, Xiaoyuan Tu, and Demetri Terzopoulos, *Cognitive modeling : Knowledge, reasoning and planning for intelligent characters*, SIGGRAPH'99, Computer Graphics Proceedings (Los Angeles) (Alyn Rockwood, ed.), Addison Wesley Longman, 1999, pp. 29–38.

[9] Lía García-Pérez and M.C. García-Alegre, *A simulation environment to test fuzzy navigation strategies based on perceptions*, 10th IEEE International Conference on Fuzzy Systems (Melbourne), December 2001.

[10] Jean-Charles Guyomarc'h, *Ethologie - seconde édition*, Abrégés, Masson, 1995.

[11] J.E.I. Hokkanen, *Visual simulations, artificial animals and virtual ecosystems*, The Journal of Experimental Biology **202** (1999), 3477–3484.

[12] David Houssin, Stefan Bornhofen, Sami Souissi, and Vincent Ginot, *Entre programmation par composants et langages d'experts - rendre la modélisation individu-centrée plus accessible à l'utilisateur*, Environnement de développement de systèmes multi-agents, vol. 21, RSTI/TSI, no. 4, 2002, pp. 525–548.

[13] Damián Isla, Robert C. Burke, Marc Downie, and Bruce Blumberg, *A layered brain architecture for synthetic creatures*, IJCAI International Joint Conference on Artificial Intelligence, Seattle, WA, 2001, pp. 1051–1058.

[14] B. Kosko, *Differential hebbian learning*, AIP Conference proceedings, vol. 151, American institute of Physics, 1986, pp. 277–282.

[15] Konrad Lorenz, *Les fondements de l'éthologie*, champs ed., Flammarion, 1973.

[16] Maja J Matarić, *Behavior-based robotics as a tool for synthesis of artificial behaviors and analysis of natural behavior*, Trends in Cognitive Science **2** (1998), no. 3.

[17] Marc Parenthoën, Patrick Reignier, and Jacques Tisseau, *Put fuzzy cognitive maps to work in virtual worlds*, Fuzz'IEEE'01 proceedings (Melbourne, Australia), vol. 1, December 2001, pp. 56–60.

[18] C. Reynolds, *Steering behaviors for autonomous characters*, Game Developers Conference (GDC) (San Jose, California), Miller Freeman Game Group, San Francisco, 1999, pp. 763–782.

[19] Craig W. Reynolds, *Flocks, herds, and schools : A distributed behavioral model*, Computer Graphics **21** (1987), no. 4, 25–34.

[20] A. Saffiotti, *Fuzzy logic in autonomous robotics : behavior coordination*, Procs. of the 6th IEEE Int. Conf. on Fuzzy Systems (1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA), IEEE Computer Society Press, 1997, pp. 573–578.

[21] Demetri Terzopoulos, Xiaoyuan Tu, and Radek Grzeszczuk, *Artificial fishes : Autonomous locomotion, perception, behavior, and learning in a simulated physical world*, Artificial Life **1** (1994), no. 4, 327–351.

[22] Nikolaas Tinbergen, *L'étude de l'instinct*, bibliothèque scientifique ed., Payot Paris, 1971.

[23] Verena M. Trenkel, Stéphanie Mahévas, Pascal Lorance, and J.F. Masset, *Evaluation of visual transects for estimating trawl efficiency of deep-sea fish species*, ICES Symposium on fish behaviour in exploited ecosystems, 2003.

[24] F. Wang and R.A. McKenzie, *Virtual Life in Virtual Environments*, Tech. Report ECS-CSG-44-98, 1998.

[25] Stewart W. Wilson, *Knowledge growth in an artificial animal*, Proceedings of the First International Conference on Genetic Algorithms and Their Applications, Lawrence Erlbaum Assoc, 1985.