Aperçu des techniques d'intelligence artificielle utilisées dans l'industrie du jeu vidéo

Dossier préparé par DS Improve SA.

Table de matières

Introduction

Cet article a pour but de donner au lecteur un bref aperçu de l'évolution des techniques d'Intelligence Artificielle[1] (IA) utilisées dans les jeux vidéo. En présentant quelques-unes d'entre elles, nous lèverons le voile sur les mécanismes qui se cachent derrière certains titres qui ont marqué l'histoire du jeu vidéo.

Depuis leur apparition dans les années 70, les jeux vidéo se sont développés à une vitesse fulgurante, encouragés par l'évolution du matériel ainsi que par la créativité des développeurs. Les progrès les plus remarquables furent principalement réalisés au niveau du réalisme visuel, des effets spéciaux et de l'environnement sonore. Les efforts consacrés au développement de l'intelligence des personnages étaient en général relégués au second plan, et ce malgré l'importance que cela peut avoir en terme d'immersion et de satisfaction des joueurs.

Pour remplir son objectif tant au niveau ludique que celui du réalisme, un jeu ne peut pas se contenter d'être agréable à l'œil. Il doit également laisser une liberté d'action au joueur tout en offrant des interactions riches et des rebondissements pour ne pas être répétitif et lassant. Il doit pouvoir adapter son niveau de difficulté en fonction des compétences du joueur.

Un jeu peut être très beau visuellement mais s'il souffre d'une IA déficiente et qu'il est mal équilibré, il ne pourra que décevoir. A l'heure actuelle, ce secteur représente tout de même un marché de plus de …40 milliard € [2]

La tendance semble heureusement s'être inversée durant ces dernières années. Les progrès techniques réalisés dans le hardware graphique grand public ainsi que la montée en fréquence et la multiplication des cœurs d'exécution au sein des processeurs ont ouvert de nouvelles perspectives (simulation physique, IA complexe), sans pénaliser le réalisme visuel (délégué aux puissantes cartes graphiques).

A présent, les développeurs de jeux vidéo suivent de près les publications académiques et la synergie entre chercheurs et développeurs est réelle à des conférences internationales[3] comme la GDC (Game Developers Conference) ou le SIGGRAPH.

Des sociétés spécialisées apparaissent également et vivent de la commercialisation de bibliothèques réutilisables basées sur l'une ou l'autre méthode issue de la recherche

[1] *le terme IA désigne ici l'ensemble des moyens utilisés pour 'diriger' les personnages contrôlés par l'ordinateur*
[2] *source : www.idate.org*
[3] *http://www.gdconf.com/, http://www.siggraph.org/*

scientifique. Des sociétés comme Kynogon, AI Implant, Spirops, SimBionic et MASA[4] commercialisent des solutions intégrables dans des moteurs de jeux vidéo mais aussi dans des simulations militaires, éducatives ou industrielles.

Techniques utilisées

Pattern movement

Le « pattern movement » est une technique qui permet de décrire de manière statique un enchaînement de mouvements simples qu'un personnage doit réaliser. C'est cette succession de mouvements qui donne l'illusion d'intelligence. Cette technique est typiquement utilisée dans les jeux d'arcade des années 80, du type *1942*[5] ou *Galaga*[6]. Dans ce style de jeu, le joueur dirige un vaisseau qui doit éliminer ses ennemis tout en les évitant ainsi que leurs tirs. Les ennemis déferlent sur le vaisseau dirigé par le joueur depuis le haut de l'écran en effectuant différents types d'approche (boucle, zigzag, looping, …).

Figure 1 : 1942 et Galaga

Prenons l'exemple de séquence suivant :

Avancer
Pivoter vers la gauche de 45°

Lorsque cette séquence de mouvement est répétée, le personnage tourne en rond. On va ainsi pouvoir décrire de manière très simple des mouvements plus ou moins complexes.

L'action sera malheureusement toujours la même et donc prévisible, quelques soient les agissements du joueur, vu que le personnage effectuera continuellement la même séquence. Une amélioration possible est de rendre les actions moins prévisibles en alternant de manière aléatoire différentes séquences.

Dans un jeu comme *Pong*[7], le « Movement pattern » ne convient plus, vu que l'adversaire doit tenir compte du joueur ou d'élément externe (dans ce cas ci la balle). L'environnement de jeu

[4] *http://www.kynogon.com/, http://www.biographictech.com/, http://www.spirops.com/, http://www.simbionic.com, http://www.masagroup.net/*
[5] *1942, 1984, Capcom*
[6] *Galaga, 1981, Namco*
[7] *Pong, 1972, Atari*

et les actions possibles étant très limités, un petit ensemble de règles simples suffit pour décrire les séquences d'actions à effectuer lors de différentes situations.

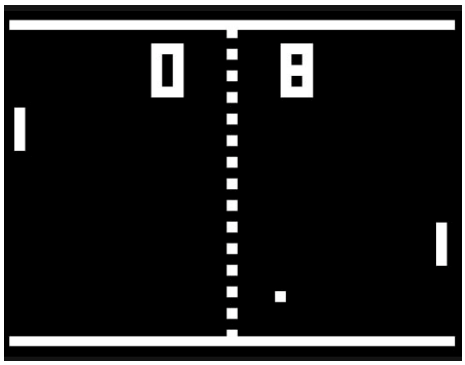

Figure 2 : Pong

Finite state machine

Dans certains FPS[8], on attribue un scénario précis à chaque personnage. Ceux-ci se comportent donc comme des acteurs de cinéma auxquels on a donné un ordre détaillé d'actions à effectuer. Si on rejoue la scène, le scénario sera toujours suivi à la lettre. Ce principe fonctionne bien pour les jeux fortement scénarisé mais limite la liberté d'action du joueur. Cela demande également un travail considérable de la part des développeurs pour décrire la marche à suivre pour chacun des acteurs.

Des jeux offrant plus de liberté aux joueurs, tels que *Half-Life*[9] ou *GTA3*[10] utilisent les Finite State Machines. Nous avons un ensemble d'états reliés entre eux par des transitions logiques. Le personnage contrôlé par l'ordinateur n'a aucune autonomie dans la prise de décision, ce sont des conditions qui activent la transition d'un état à un autre.

Par exemple, un soldat faisant son tour de garde pourrait être modélisé de la façon suivante.

Figure 3 : Exemple de machine à état fini

En fonction des conditions, l'état du garde va pouvoir évoluer au cours du temps. De l'état « monter la garde », il va passer à « Combattre » lorsqu'il repère un ennemi, par exemple.

Un grand avantage de cette technique est qu'elle est relativement simple à modéliser et accessible à des non-programmeurs. Il est possible d'utiliser une FSM à plusieurs niveaux pour

[8] *FPS : first person shooter*
[9] *Half-Life : 1998, Valve Software*
[10] *GTA3 : 2001, DMA Design*

affiner les actions. Dans notre exemple, l'action « Combattre » pourrait être décrite, elle-même, par sa propre FSM, on parle alors de « hierarchical finite state machine » ou HFSM.

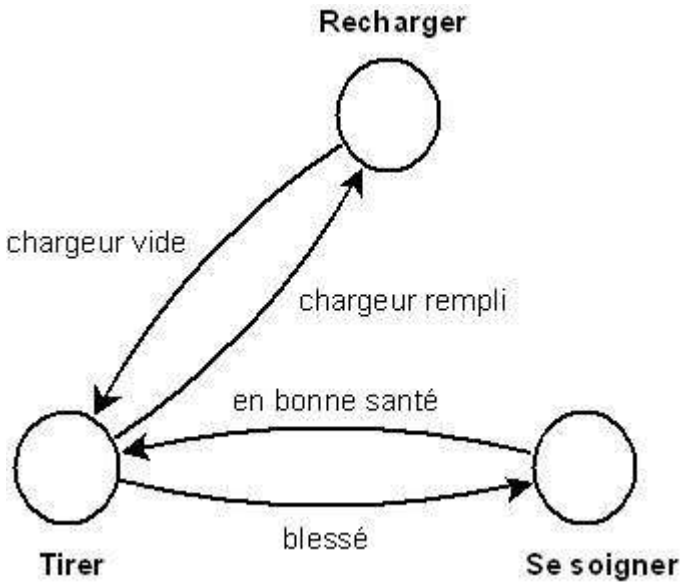

Figure 4 : FSM de l'action « combattre »

Cette technique des HFSM est appréciée dans l'industrie du jeu vidéo par son formalisme simple et sa modularité (on peut récupérer des sous-FSM, qui représente des compétences particulières, dans différentes HFSM et donc dans les comportements de différentes créatures). Elle est utilisée dans différents jeux récents, dont notamment le jeu *Destroy All Humans*[11].

Figure 5 : Destroy All Humans

[11] *Destroy All Humans : Pandemic Studios, 2005*

Smart Terrain / Object

Dans certains jeux, des informations sur la topologie de l'environnement (par ex : « point de passage, « zone à couvert ») sont injectées dans le terrain. Quand un personnage se trouve dans une zone, il peut en extraire les informations.

Poussé à son extrême, ce principe nous mène au concept de Smart Object. En utilisant ce concept, le personnage est représenté de manière relativement simple. Ce sont les objets et les lieux présents dans son environnement qui vont lui donner des instructions (sous forme de message ou de scripts) sur ce qu'il peut faire ou non. Notre personnage ne doit pas apprendre à se servir d'un objet, c'est l'objet qui lui dit comment l'utiliser.

Cette approche à été notamment utilisée pour le jeu *The Sims*[12]. Dans ce programme, le personnage a des besoins et des envies relativement simples (boire, manger, se reposer, s'amuser, …), modélisés par une Finite State Machine. Les objets sont là pour combler les besoins et les envies du personnage et le font savoir, en émettant dans un certain périmètre, des messages, à propos de ce qu'ils offrent ou permettent.

Prenons l'exemple suivant : un personnage, ayant faim, se trouve en présence d'un plat préparé et d'un four. Le plat va émettre comme message qu'il doit être cuit pour pouvoir rassasier notre personnage. Le four va, quant à lui, émettre comme message qu'il peut cuire les aliments. Notre personnage utilisera d'abord le four pour cuire son aliment. Le comportement du personnage sera donc dictée par le four (ouvrir le four, placer le plat à l'intérieur, fermer le four, allumer le four, …). Une fois la séquence finie, le message émis par le plat sera « je peux être manger » et notre personnage va pouvoir enfin se régaler.

Figure 6 : The Sims 2

La richesse du comportement se trouve donc distribuée au sein des objets et non dans le personnage. Le personnage sait toujours comment interagir avec son environnement et ses comportements sont facilement extensibles puisque ce sont les objets qui suggèrent des actions.

[12] *The Sims : 2000-2005, Maxis*

Logique floue

La logique floue ou « fuzzy logic » est une technique issue des milieux académiques dans les années 60 et basée sur la théorie des ensembles flous. Elle a depuis été utilisée dans de nombreuses applications industrielles : en automatisme (freins ABS), dans le contrôle aérien, dans le diagnostic médical, etc … et dans les jeux vidéos.

L'idée sous-jacente est de permettre à une condition d'être dans un autre état que *vrai* ou *faux*. Autrement dit, la nécessité de trouver une formalisation mathématique pour exprimer des notions vagues et approximatives.

Dans un jeu vidéo, cela permet d'utiliser des opérations logiques sur des valeurs imprécises (« assez loin », « plutôt dangereux ») et de déclencher simultanément plusieurs états, à des niveaux d'importance différents (« faché », mais davantage « peur »), ce qui permet au final d'obtenir des comportements moins tranchés. Le jeu *Sims* utilise notamment ce genre de technique en complément aux smart terrains (voir chapitre précédent).

Les règles de fonctionnement d'un joueur de football peuvent, par exemple, s'exprimer de la façon suivante :

SI goal *proche* ET adversaire NON *proche* : **Tirer au but**
SI adversaire *proche* ET coéquipier NON *loin* : **Passer le ballon**
SI adversaire *proche* ET coéquipier *loin* : **Dribbler**

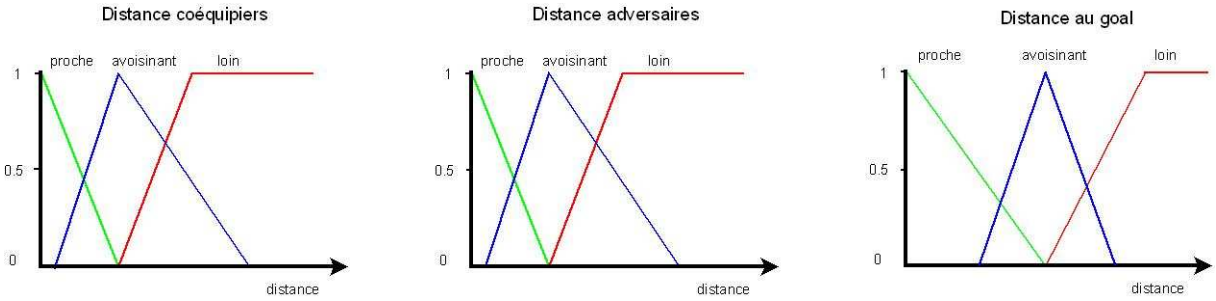

Figure 7 : Représentation graphique de fonctions d'appartenance

Les notions *proche*, *avoisinant* ou *loin* sont déterminée par un « degré d'appartenance » à un ensemble. Cette valeur comprise entre 1 et 0 est calculée au moyen de fonctions d'appartenance. Le système d'inférence peut produire ensuite, sur base de ces règles et des évènements et stimuli perçus dans l'environnement, une cotation des activités qui peut être la suivante :

Tirer au but avec une valeur de 0.1
Passer le ballon avec une valeur de 0.2
Dribbler avec une valeur de 0.7

L'action à effectuer est simplement celle ayant la meilleure cote. Des extensions de ce système permettent d'utiliser des filtres pour choisir l'action gagnante.

Modèles inspirés par la biologie

Dans leur quête pour reproduire une imitation de l'intelligence humaine, les chercheurs en intelligence artificielle se sont inspirés de différents mécanismes biologiques pour concevoir des modèles informatiques de décision.

Dans le domaine du jeu vidéo, deux de ces méthodes ont été utilisées avec succès : les réseaux de neurones artificiels et les algorithmes génétiques.

Les neurones artificiels tentent de reproduire un réseau artificiel de neurones, interconnectés par des synapses. Techniquement, ces neurones sont organisés en couches successives, chaque neurone d'une couche prenant ses entrées dans des neurones de la couche précédente. Des stimuli externes excitent la première couche et par propagation, l'excitation se propage de couche en couche pour aboutir au final à des valeurs de sortie, sur base desquelles sera prise la décision finale. Cette technique permet de prendre des décisions s'appuyant davantage sur la perception que sur le raisonnement.

A chaque synapse est associé un poids qui amplifie le signal perçu. Les signaux d'entrée sont combinés en tenant compte des poids des synapses, et le tout est fourni à une fonction d'activation qui est non linéaire et détermine la valeur d'activation du neurone. Celle-ci est ensuite utilisée comme stimulus d'entrée pour les neurones de la couche suivante.

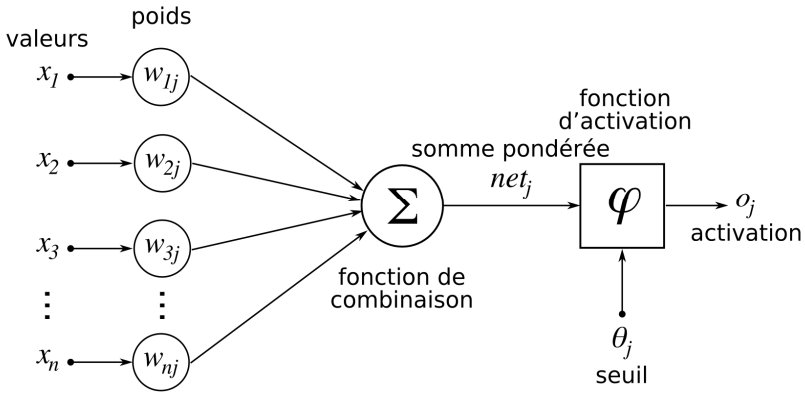

Figure 8 : Neurone artificiel : combinaison du signal perçu (valeurs x d'entrée) et activation non linéaire du neurone.

En fonction des types de réseau, différentes fonctions de combinaison peuvent être utilisées : fonction de combinaison sous forme de produit scalaire entre les entrées et les poids synaptiques, fonction de combinaison basée sur la distance euclidienne entre les entrées, …

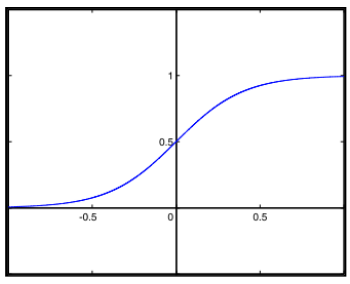

Figure 9 : Fonction de type sigmoïde

En ce qui concerne la fonction d'activation, l'idée est utiliser un seuil en dessous duquel, le neurone est inactif (0 ou – 1) et au-dessus duquel, le neurone est actif (1). Aux alentours du seuil, on a une phase de transition. Les fonctions mathématiques de type sigmoïde ou de type hyperbolique offrent ces propriétés. D'autres fonctions non linéaires présentant un seuil sont également possibles.

Toute la finesse dans l'utilisation des réseaux de neurones, réside dans le nombre de couches et de neurones à utiliser, mais surtout dans la technique utilisée pour l'apprentissage ou ajustement des différents paramètres (coefficients des synapses) de façon à corriger les erreurs commises. L'apprentissage peut être supervisé (on force le réseau à converger vers un état final précis quand on lui présente des stimuli) ou être non-supervisé (on le laisse converger librement vers n'importe quel état lorsqu'on lui présente des stimuli).

L'apprentissage supervisé a notamment été utilisé dans différents jeux, notamment les jeux de voitures (*Colin McRae Rally*[13]) ou les jeux de combats (*Tekken*[14]). L'apprentissage consiste ici à faire des parcours de circuit ou des séances de combat manuellement, et de corriger ensuite les coefficients des synapses, en utilisant la technique de rétro-propagation. C'est-à-dire en rétro-propageant l'erreur commise par un neurone à ses synapses et aux neurones qui y sont reliés (les poids synaptiques qui contribuent à engendrer une erreur importante sont modifiés de façon plus significative que ceux qui ont engendrés une erreur plus faible). D'autres jeux ont également utilisé ces techniques de réseaux neuronaux pour des simulations de vie artificielle comme *Creatures*[15] et *Black & White*[16].

Figure 10 : Creatures, un jeu utilisant les réseaux de neurones et simulant la vie artificielle des Norns

Les algorithmes génétiques sont utilisés quant à eux dans différents problèmes d'optimisation, afin de déterminer des solutions qui sont satisfaisantes sans être pour autant optimales. Les chercheurs en intelligence artificielle se sont ici inspirés des mécanismes de sélection naturelle dans la théorie de l'évolution.

Dans un jeu vidéo, cela revient, par exemple, à ramener les caractéristiques d'un opposant sous la forme d'une chaîne d'ADN artificielle, et d'effectuer une sélection parmi les meilleurs

[13] *Colin McRae Rally serie*, 1998-2007, *CodeMasters*
[14] *Tekken serie*, 1994-2007, *Namco*
[15] *Creatures*, 1996, *Millenimum Interactive*.
[16] *Black and White*, 2001, *Lionhead Studios*

opposants, pour ensuite les croiser et obtenir le meilleur des caractéristiques d'origine. De temps à autre, des mutations sont effectuées sur cette chaîne d'ADN pour obtenir de nouvelles variantes.

La difficulté est donc ici de modéliser les caractéristiques sous forme de chaîne d'ADN artificielle représentées par des bits ou des réels, de déterminer une fonction adéquate pour sélectionner les meilleurs candidats et de développer ces méthodes de mutation et de croisement. Plusieurs techniques de sélection sont utilisées : sélection par rang (choisir les individus possédant les meilleurs scores d'adaptation), sélection proportionnelle à l'adaptation (principe de la roue de la fortune biaisée ou chaque individu est représenté par une portion proportionnelle à son adaptation), sélection par tournoi (choisir la sélection proportionnelle sur des paires d'individus, et choisir ensuite parmi ces paires, l'individu qui a le meilleur score d'adaptation), sélection uniforme (choix aléatoire sans tenir compte de la valeur d'adaptation).

La technique, dite des systèmes à classeurs (ou classifier systems), a été expérimentée dans une version modifiée du jeu *Half Life*[17]. Dans sa version simple, cette technique consiste à modéliser des règles de déclenchement d'actions sous forme de patterns de 0, 1 et * (0 ou 1). Une action se déclenche si son pattern coïncide avec les stimuli présentés en entrée. Ici, des algorithmes génétiques ont été utilisés pour sélectionner les meilleurs patterns (en cotant les actions effectuées).

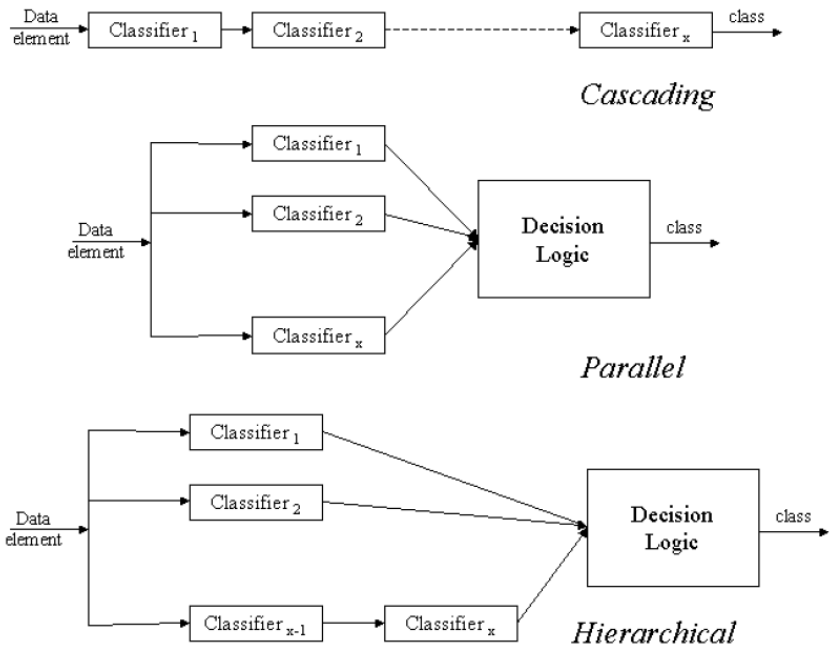

Figure 11 : Organisation des systèmes à classeurs

Cette expérience a permis d'optimiser automatiquement le comportement d'une équipe de quatre bots. Leurs performances se sont avérées être au final comparable ou supérieure à celle des meilleurs bots disponibles sur Internet (scriptés manuellement).

[17] *Half Life*, 1998, *Valve Software*, version modifiée : http://animatlab.lip6.fr/KodamatMainFr

Figure 12 : Half Life modifié pour tester les systèmes à classeurs

Modèles inspirés par l'éthologie

Certains chercheurs en intelligence artificielle se sont inspirés de l'éthologie, c'est-à-dire des études comportementales effectuées sur les animaux, afin de reproduire des êtres artificiels. L'accent est plutôt mis sur l'émotion et le réalisme plutôt que sur le comportement réfléchi.

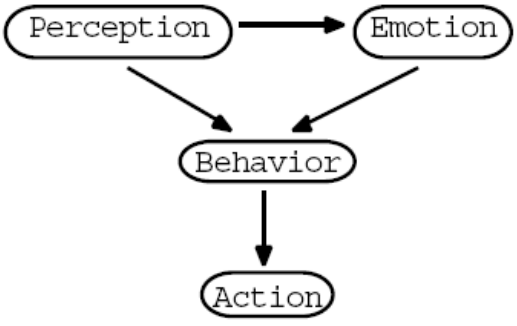

Figure 13 : Structure d'un modèle basé sur le comportement

En général, le comportement de l'animal est modélisé comme une hiérarchie de comportements plus spécifiques. A chaque niveau, a lieu une compétition entre les comportements pour prendre le contrôle de l'animal et ainsi imposer son action.

Figure 14 : Nintendogs de Nintendo

Au niveau des stratégies de compétition, deux principaux modèles ont été utilisés : *inhibition et fatigue* (les comportements tentent de s'inhiber mutuellement, et lorsqu'un comportement a le contrôle, il perd progressivement de sa pertinence dû à un phénomène de fatigue), et *free-flow* (on fait propager une quantité d'énergie au sommet de la hiérarchie, et les comportements ayant absorbés le plus d'énergie seront les vainqueurs).

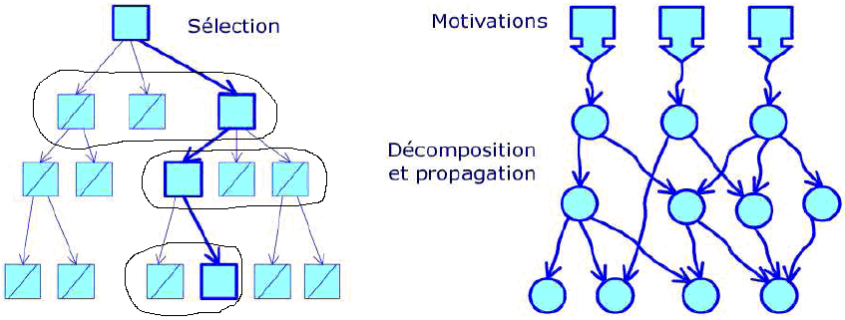

Figure 15 : Inhibition-Fatigue vs Free-Flow

Ces types de modèles ont été utilisés dans le projet de chien robotique *AIBO*[18] de Sony et le jeu *Nintendogs* de Nintendo pour sa console à écran tactile Nintendo DS.

Figure 16 : Le chien robotique AIBO de Sony

[18] http://support.sony-europe.com/aibo/

Le Futur ?

Difficile à pronostiquer, mais l'augmentation des capacités mémoire et l'arrivée chez les particuliers des processeurs massivement multi-coeurs et des cartes graphiques dotées de processeurs spécialisés pour le traitement parallèle de données, permettront sans doute des implémentations en temps réel peu gourmande pour toute une série d'algorithmes déjà publiés par le passé mais qui n'étaient pas exploitables en condition réelle dans un jeu vidéo. Cela peut aller de l'implémentation de réseaux de neurones complexes à des moteurs d'inférence de logique floue utilisant une base de règles importantes.

Mais la puissance de calcul sera utilisée dans les jeux pour renforcer également d'autres aspects complémentaires, comme l'amélioration du comportement apparent des personnages.

En effet, le réalisme du comportement d'un personnage ne dépend pas seulement de ses choix d'actions, mais également de la richesse de ses mouvements et de ses expressions faciales, tout comme la simulation réaliste de ses sens. La conversation avec les avatars fait aussi partie de la crédibilité (voir *AliceBot*[19]) et les progrès réalisés en matière de synthèse vocale (ex. *Acapela, Loquendo* [20]) offrent des perspectives intéressantes.

Figure 17 : Euphoria (NaturalMotion)

Au niveau du réalisme, des progrès ont lieu avec l'arrivée des moteur d'animations procédurales physiques en temps-réel (ex. *Euphoria*[21]), des modèles d'expression faciales (ex. *EMotion FX* facial animation[22]), et la simulation des sens en tenant compte des phénomènes d'occlusion, de réflexion et de réfraction de la lumière, de l'influence de l'éclairage, de la propagation des odeurs ou des propriétés acoustiques de l'environnement.

Les progrès en matière de puissance de calcul permettront un renforcement global du réalisme en combinant ces différents éléments.

[19] http://www.alicebot.org/
[20] http://demo.acapela-group.com/, http://actor.loquendo.com/actordemo/
[21] http://www.naturalmotion.com/euphoria.htm
[22] http://www.mysticgd.com

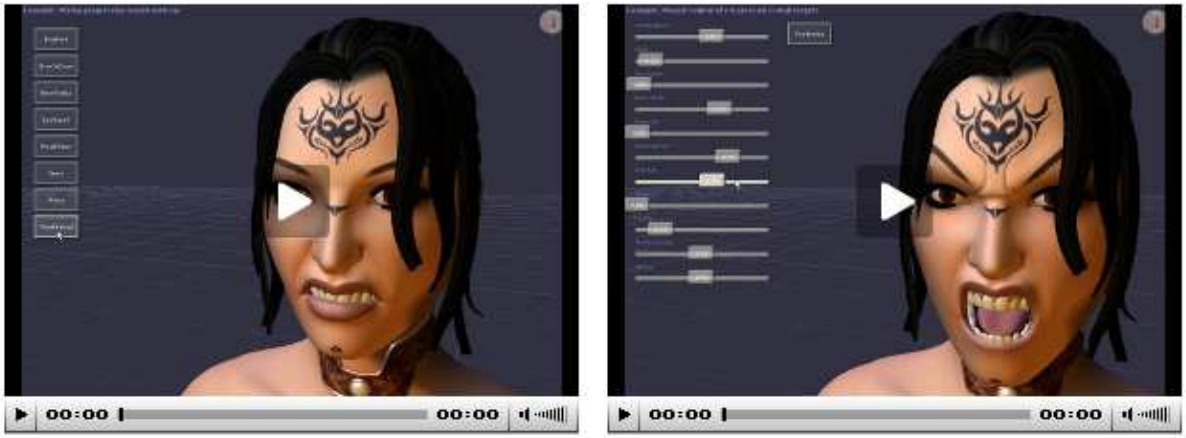

Figure 18 EMotion FX (Mystic Game Development)

Qui est DS Improve SA ?

DS Improve SA[23] est une société de services informatiques basée à Bruxelles, dont le champ d'activités est large (conception de sites internet, développement logiciel sur mesure, consultance spécialisée et expertise technique, solutions intégrées à valeur ajoutée, recherche industrielle et multimédia).

Laurent Lardinois est diplômé de l'Université Libre de Bruxelles en Informatique. Il exerce le rôle de Software Architect, spécialisé dans le développement système, industriel et dans les technologies utilisées dans le monde du jeu vidéo. Il est responsable de l'unité R&D de DS Improve SA.

Dimitri Duchateau est diplômé de l'Université Libre de Bruxelles en Informatique. Il a réalisé son travail de fin d'étude dans le domaine des MMORPGs (jeux en lignes massivement multi-joueurs) et travaille comme Software Developer au sein de l'équipe R&D de DS Improve SA.

Vous pouvez les joindre en visitant la page R&D sur le site http://www.dsimprove.be

Références

David M. Bourg & Glenn Seemann - AI for Game Developers - O'Reilly 2004 - ISBN: 0596005555

Steve Rabin - AI Game Programming Wisdom – Charles River Media 2002 - ISBN: 1584500778

Steve Rabin - AI Game Programming Wisdom 2 – Charles River Media 2003 - ISBN: 1584502894

John Krajewski - "Creating All Humans" – in Game Developer Magazine – December 2006 – CMP United Business Media

[23] http://www.dsimprove.be

Yifan Li, Petr Musilek and Loren Wyard-Scott, "Fuzzy Logic in Agent-Based Game Design", Fuzzy Information, 2004. Processing NAFIPS '04. Volume: 2, On page(s): 734- 739 Vol.2, Department of Electrical and Computer Engineering, University of Alberta, 2004.
http://www.ece.ualberta.ca/~wyard/papers/NAFIPS2004.pdf

Darryl Charles and Michaela Black, "Dynamic Player Modelling; A Framework for Player-centred Digital Games", in Proceedings of CGAIDE 2004 International Conference on Computer Games, School of Computing and Information Engineering, University of Ulster, Northern Ireland, 2004.
http://info200.infc.ulst.ac.uk/~darryl/Papers/CGAIDE04/DynamicPlayerModelling.pdf

Adriaan G.Tijsseling, "Sequential Information Processing using Time-Delay Connections in Ontogenic CALM networks", in Neural Networks, IEEE Transactions, Jan 2005, Volume 16, Issue 1, pages 145-159, Cognitive Neuroinformatics Group, Neuroscience Research Institute, Tsukuba, Japan, 2005.
http://kung-foo.tv/download/ieee.pdf

Berthouze, Luc and Tijsseling, Adriaan, "A Neural Model for Context-dependent Sequence Learning. Neural Processing Letters", 23 (1). pp. 27-45. ISSN 1370-4621, 2006
http://staff.aist.go.jp/luc.berthouze/Papers/journals/npl06.pdf

Bruce Blumberg, "Action-selection in hamsterdam: Lessons from ethology". In Cliff, D., Husbands, P., Meyer, J.-A., and S, W., editors, From Animals to Animats: The Third International Conference on Simulation of Adaptive Behavior, pages 108-117. The MIT Press, Cambridge, MA, 1994.
http://www.msci.memphis.edu/~classweb/comp7990/fall2002/agents/slides/HAM.ppt

Arkin, R.C.; Fujita, M.; Takagi, T.; Hasegawa, R., "Ethological modeling and architecture for an entertainment robot", Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on Volume 1, Issue , 2001 Page(s): 453 - 458 vol.1, 2001
http://www.cc.gatech.edu/ai/robot-lab/online-publications/icra-2001-sony.pdf

Arkin R.C.1; Fujita M.; Takagi T.; Hasegawa R., "An Ethological and Emotional Basis for Human-Robot Interaction", Robotics and Autonomous Systems, Volume 42, Number 3, 31 March 2003 , pp. 191-201(11), 2003
http://www.cc.gatech.edu/ai/robot-lab/online-publications/sony-iros.pdf

Becheiraz, P. ; Thalmann, D., "A Behavioral Animation System for Autonomous Actors personified by Emotions", Proc. Workshop on Embodied Conversational Characters, WECC 98. (1998), LIG-Comput. Graphics Lab., Swiss Fed. Inst. of Technol., Lausanne, Switzerland, 1998
http://ligwww.epfl.ch/Publications/pdf/Becheiraz_Thalmann_WECC_98.pdf

D.Thalmann, S. R. Musse and M. Kallmann, "Virtual Humans' Behaviour: Individuals, Groups, and Crowds", invited paper, Proceedings of the International Conference on Digital Media Futures, British Computer Society, Bradford, UK, April 13-15, 1999
http://ligwww.epfl.ch/~thalmann/papers.dir/Bradford.pdf

Romesh Ranawana, Vasile Palade, "Multi Classifier Systems – A Review and Roadmap for Developers", to appear in March 2006 issue of the Journal of Hybrid Intelligent Systems, IOS Press Amsterdam, 2005.
http://oxford.academia.edu/documents/RomeshRanawana_RP2005b.pdf

Robert, G., Portier, P., and Guillot, A. "Classifier systems as 'animat' architectures for action selection in MMORPG". In Mehdi, Q., Gough, N., and Cavazza, M., editors, Game-on 2002, pages 121-125, 2002.
http://animatlab.lip6.fr/papers/gameon.pdf