Chapitre 1

Introduction

1.1. Contexte

Ce livre traite de l'Intelligence Artificielle (IA) dans les jeux vidéo et les jeux de réflexion.

Les jeux sur ordinateur et consoles prennent de plus en plus d'importance dans notre société. Ainsi l'industrie du jeu vidéo a connu une forte croissance pendant plusieurs années, et la taille de son marché mondial dépasse désormais les vingt milliards de dollars.

On a assisté depuis les débuts des jeux vidéo à une complexification du comportement des personnages non joueurs. Des comportements stéréotypés et prévisibles des premiers jeux vidéo, on est passé à des comportements complexes qui font aussi appel à la coordination des entités.

Les joueurs sont demandeurs de comportements réalistes de la part des personnages non joueurs, et pour des jeux complexes cette partie comportement est difficile à mettre en œuvre. La programmation de l'IA est pourtant souvent traitée à la fin de la programmation d'un jeu vidéo, ce qui la pénalise par rapport aux autres éléments du jeu. Il faut toutefois préciser que les tests de l'IA ne peuvent souvent être fait que lorsque le jeu est déjà bien avancé.

La pertinence des choix de l'IA dans un jeu vidéo est très liée à la satisfaction du joueur. Des comportements visiblement inadaptés de la part des personnages non

---

Chapitre rédigé par Tristan CAZENAVE.

joueurs rendent un jeu bien moins intéressant. Il est d'ailleurs courant de voir les joueurs expérimentés se lasser des comportements de l'IA d'un jeu et se tourner vers le jeu en réseau dans lequel ils peuvent rencontrer d'autres joueurs humains.

D'un point de vue académique, plusieurs domaines de l'Intelligence Artificielle qui ont été développés pour d'autres applications que les jeux peuvent y trouver des applications. Les recherches en apprentissage, en planification ou en systèmes multi-agents sont ainsi tout à fait applicables au cas de la prise de décision dans les environnements complexes que sont les jeux.

Un autre aspect de la programmation des jeux est le temps réel. Dans les jeux vidéo, certaines décisions doivent être prises en moins d'un centième de seconde. Dans les jeux de réflexion, l'aspect temps réel est aussi présent puisque les temps de réflexion et la mémoire sont limités, même si les temps de réponse sont nettement supérieurs à ceux des jeux vidéo.

Les jeux vidéo, comme les jeux classiques offrent un terrain privilégié d'expérimentation pour l'IA. Il est en effet simple dans les deux cas de faire jouer entre elles des IA qui utilisent des approches différentes, et ainsi de les comparer.

L'IA dans les jeux vidéo offre un large panorama de recherches possibles. On peut suivre John Laird dans sa classification des différents sujets de recherche en IA ayant trait aux jeux vidéo [LAI 01]. Il donne pour chaque type de jeu les sujets de recherche qui s'y rapportent. Il donne aussi les différents rôles que peut tenir l'IA dans un jeu, à savoir l'ennemi tactique comme dans les jeux d'action, le partenaire, les personnages, les ennemis stratégiques dans les jeux de stratégie temps réel, l'unité qui obéit aux ordres de haut niveau et le commentateur. Michael Buro se focalise sur les jeux de stratégie temps réel et propose de travailler en priorité sur la planification avec adversaire et incertitude, l'apprentissage, la modélisation de l'adversaire, et les raisonnements spatiaux et temporels [BUR 04].

Dans les jeux de réflexion à information complète, les progrès effectués dans les algorithmes de recherche arborescente et d'analyse rétrograde permettent aux programmes d'égaler, ou de surpasser les meilleurs joueurs humains dans de nombreux jeux classiques comme les Echecs, les dames anglaises [SCH 97] ou encore l'Awele [ROM 03]. Certains jeux classiques résistent toutefois encore aux algorithmes de recherche. C'est par exemple le cas du jeu de Go, qui est combinatoirement difficile avec une moyenne de deux cent cinquante coups possibles, et pour lequel il est difficile de construire une fonction d'évaluation qui soit à la fois pertinente et rapide [BOU 01].

Dans les jeux de réflexion à information incomplète, certains programmes ont des niveaux proches des meilleurs humains comme TD-Gammon au Backgammon [TES 95] ou Maven qui dépasse les meilleurs joueurs de Scrabble [SHE 04]. D'autres ont des niveaux de joueurs intermédiaires comme Poki au Poker [BIL 03] ou Jack au

Bridge. Les algorithmes utilisés sont différents des jeux à information complète et on est plutôt là dans le domaine des simulations de Monte-Carlo ou de l'apprentissage par renforcement. Notons tout de même que l'Alpha-Béta peut être utilisé pour résoudre des donnes ouvertes, et que les algorithmes de Monte-Carlo sont utilisés avec succès au jeu de Go, il y a donc des passerelles entre les jeux à information complète et incomplète.

1.2. Contenu du livre

Les premiers chapitres traitent de l'apprentissage dans les jeux. Les trois premiers utilisent les jeux vidéo alors que les deux suivants sont appliqués à des jeux de réflexion. La transition vers la planification au sens large se fait à travers les jeux de réflexion puisque le chapitre suivant traite d'un algorithme de recherche appliqué à la résolution d'Atarigo. Les quatre derniers chapitres traitent de la planification dans les jeux vidéo.

Voici plus en détail le contenu des différents chapitres dans l'ordre dans lequel ils apparaissent :

William Tambellini, Cédric Sanza et Thomas Busser font une présentation générale de travaux ayant trait à la génération automatique et semi automatique de comportements pour les personnages de jeux vidéo. Ils présentent quatre techniques en évitant volontairement les réseaux de neurones jugés peu modifiables et peu compréhensibles. Ces quatre techniques sont les algorithmes génétiques, les systèmes de classeurs, les arbres de décision et l'apprentissage par renforcement. Ils montrent l'application de chaque technique à des jeux vidéo. On découvre ainsi comment est implémentée l'animation d'un serpent dans un environnement composé d'obstacles et d'une cible mobile, à l'aide d'algorithmes génétiques. Ils montrent aussi comment on peut apprendre des comportements pour une simulation de football appelée NeViS à l'aide de systèmes de classeurs, comment réaliser des agents d'une simulation de basket-ball à l'aide de systèmes de classeurs génériques, comment imiter un joueur humain dans un monde virtuel intégrant des ennemis hostiles en utilisant des arbres de décision. Ils montrent enfin comment l'apprentissage par renforcement a été utilisé dans le jeu Before Christ. Ils abordent en conclusion les caractéristiques que devrait avoir l'Intelligence Artificielle d'un jeu vidéo, à savoir la rapidité, la lisibilité, la modifiabilité et le niveau de jeu en relation avec le plaisir du joueur.

Gabriel Robert et Agnès Guillot décrivent comment l'approche Animat ou Intelligence Artificielle située peut s'appliquer aux jeux vidéo. Ils utilisent pour cela une architecture motivationnelle et hiérarchique à base de systèmes de classeur (MHiCS). Cette architecture est appliquée à un jeu de capture de drapeau de Team Fortress Classic. Elle est présentée comme une alternative viable à l'architecture classique à base

de scripts. Une propriété importante de cette approche est qu'elle effectue un apprentissage en ligne au cours de chaque partie. L'architecture est décomposée en niveaux, chaque niveau utilisant le précédent pour prendre ses décisions. Le premier niveau est celui des motivations, qui peuvent être le score collectif, le score individuel ou la survie. Viennent ensuite, au deuxième niveau, les systèmes de classeurs, puis au troisième niveau les actions activables et enfin les ressources d'action. L'architecture est en interaction avec l'environnement et les senseurs. Elle est testée contre l'équipe FoxBot qui est une équipe de joueurs artificiels ayant un niveau de joueur confirmé, et qui fait partie des meilleurs joueurs artificiels. L'utilisation de classeurs fixes ne permet pas de battre l'équipe FoxBot, cependant en ajoutant l'apprentissage, l'architecture dépasse le niveau de l'équipe FoxBot, et le dépasse encore plus nettement en combinant apprentissage et création de règles. Les auteurs évaluent enfin l'apprentissage, les motivations, le temps d'évaluation et la création de règles. Ils concluent en évoquant les caractéristiques de leurs bots qui apparaissent plus crédibles, mieux coordonnés et moins agressifs que les FoxBots, ainsi qu'en attestant de l'efficacité des systèmes de classeurs pour un univers complexe.

Samuel Manier a programmé une Intelligence Artificielle pour le jeu d'action multijoueur à la première personne Wolfenstein : Enemy Territory™. Il utilise pour cela une navigation par waypoints et une recherche de plus court chemin avec A*, un réseau de neurones pour contourner les obstacles, ainsi qu'un algorithme génétique pour entraîner le réseau de neurones. L'intelligence artificielle choisit parmi plusieurs actions en fonction de leurs priorités, ces priorités sont calculées à l'aide d'un ensemble de règles floues. L'architecture générale de l'intelligence artificielle est alors présentée. Le bot qui implémente cette intelligence artificielle est distribué gratuitement sur Internet et y rencontre un vif succès.

Bruno Bouzy et Guillaume Chaslot utilisent l'apprentissage par renforcement pour améliorer le niveau de jeu d'un programme de Go. Indigo, le programme de Go utilisé comme base d'expérience, a une architecture Monte-Carlo. Cela signifie qu'il effectue une grand nombre de parties pseudo-aléatoires à partir de la position que l'on veut évaluer. L'utilisation de simulations de Monte-Carlo au Go est une réponse intéressante au problème difficile de la construction d'une fonction d'évaluation pour ce jeu. L'évaluation d'une position revient dans cette architecture à effectuer la moyenne des résultats des parties pseudo-aléatoires qui commencent à la position à évaluer. L'algorithme de Monte-Carlo classique choisit les coups à jouer avec une égale probabilité dans les parties pseudo-aléatoires. Une amélioration de cet algorithme est de biaiser le choix des coups de façon à choisir plus souvent ceux qui sont réputés être meilleurs. Pour détecter les bons coups, on peut utiliser des règles simples. Ces règles simples sont associées à des urgences qui sont utilisées pour biaiser le choix des coups. L'apprentissage consiste à trouver de bonnes urgences pour les règles. Une fois les urgences apprises, le programme est meilleur que le programme Monte-Carlo classique mais reste moins bon qu'un programme dont les urgences ont été affinées à la main.

Bernard Helmstetter et Tristan Cazenave abordent les thèmes de l'apprentissage et de la recherche heuristique. Ces méthodes sont appliquées à Lines of action, un jeu de réflexion à deux joueurs et à information complète. BING, le programme décrit, a terminé deux fois deuxième aux olympiades des ordinateurs sur des scores très serrés et a battu YL le vainqueur des précédentes olympiades. Les heuristiques de recherche utilisées sont choisies parmi les heuristiques classiques déjà implémentées dans GNU Chess. De même les structures de données liées à la représentation du damier sont reprises de GNU Chess. En ce qui concerne la fonction d'évaluation, plusieurs composantes plus ou moins classiques pour Lines of action sont calculées rapidement. Le calcul incrémental du nombre d'Euler permet de détecter très rapidement les victoires. La méthode d'apprentissage utilisée est la méthode des différences temporelles, le réseau de neurone utilisé est très simple et l'apprentissage se fait très rapidement sur une base de parties que le programme a joué contre lui même. En résumé ce chapitre montre comment écrire rapidement un bon programme de Lines of Action.

Frédéric Boissac et Tristan Cazenave présentent des heuristiques de recherche qui permettent d'accélérer fortement la résolution d'Atarigo. Atarigo est une simplification du jeu de Go dans lequel le premier qui capture une chaîne adverse a gagné. L'algorithme présenté est le premier à résoudre Atarigo 6×6 avec un goban vide, et Atarigo 7×7 avec quatre pierres croisées. L'algorithme est implémenté avec le programme de Go et d'Atarigo Dariush. Les heuristiques utilisées sont au nombre de sept. Certaines sont classiques comme les tables de transposition ou les coups qui tuent. D'autres sont nouvelles, comme la limitation du nombre de coups pour le joueur qui maximise la fonction d'évaluation, ou constituent une variante d'heuristiques déjà connues, comme la détection de gain par un *alpha-beta* secondaire ou la détection de menaces à l'aide de ce même *alpha-beta* secondaire. La résolution d'Atarigo 6×6 avec quatre pierres croisées est elle aussi fortement améliorée par rapport aux résultats précédemment publiés.

Tristan Cazenave compare ensuite différentes stratégies pour un jeu de stratégie temps réel simple. Il décrit des stratégies simples et statiques qui s'appliquent de la même manière quelle que soit la situation, et des stratégies dynamiques qui testent différentes stratégies sur la situation courante à l'aide de simulations, et choisissent la meilleure. Les stratégies dynamiques utilisent aussi bien des simulations de Monte-Carlo qui évaluent des buts tactiques, que des simulations de rencontres de stratégies statiques. Toutes les stratégies sont évaluées en jouant des parties les unes contre les autres sur différentes cartes. La stratégie qui ressort comme la meilleure est la stratégie dynamique qui simule à chaque pas de la partie tous les combats entre stratégies statiques, et retient et applique celle qui est en moyenne la meilleure.

Dans leur chapitre, Armelle Prigent, Ronan Champagnat et Pascal Estraillier nous décrivent comment faire évoluer le scénario d'un jeu en utilisant la logique linéaire et les réseaux de Petri. Dans les jeux pour lesquels un unique scénario n'est pas défini à l'avance, il faut faire évoluer le scénario en fonction de la situation du joueur

et de l'ensemble du jeu. L'évolution du scénario répond à plusieurs contraintes qui permettent de conserver la cohérence, la validité et l'intérêt du jeu. L'originalité de leur approche est d'effectuer la scénarisation dynamiquement, ce qui permet de ne pas engendrer statiquement tous les scénarios possibles.

Damien Devigne, Philippe Mathieu et Jean-Christophe Routier proposent un modèle centré interaction. C'est un environnement de planification en chaînage arrière appliqué à la gestion d'équipes d'agents dirigées par un chef. Il satisfait plusieurs contraintes des simulations par agents. Il permet de réutiliser des composants, de s'intéresser au déroulement de la simulation et non pas seulement à son résultat, de prendre en compte l'aspect situé en incluant les déplacements au niveau du raisonnement des agents et pas seulement au niveau de l'exécution des plans, mais aussi de planifier des comportements d'équipe. Les interactions entre agents sont décrites, ainsi que leur représentation, puis viennent la description des agents, de l'environnement, de l'architecture des agents, et de leur cycle de fonctionnement. Les ordres sont donnés aux agents sous formes de buts que l'agent doit chercher à atteindre. Pour cela il utilise la planification en prenant en compte les déplacements. Lorsque l'environnement change, l'agent peut replanifier partiellement ses buts. Les auteurs décrivent ensuite la planification dans une équipe. Celle-ci commence par la planification du chef qui délègue ensuite une partie de la planification aux autres agents. Les affectations des actions sont effectuées par transformation du problème d'affectation en problème de flot maximal dans un graphe. Le modèle est testé sur un jeu dynamique dans lequel deux agents sont en concurrence pour prendre un objet dans un labyrinthe. Dans ce jeu les positions et les propriétés des objets ne sont pas immuables ce qui montre le fonctionnement du module de mise à jour des plans.

Karim Sehaba et Pascal Estraillier se placent dans le cadre du contrôle d'exécution des applications interactives, en particulier les jeux. Leur objectif consiste à définir un modèle permettant d'analyser le comportement du joueur à partir des données hétérogènes provenant de différentes sources. Il s'agit des évènements générés par les actions physiques du joueur. Ces évènements sont le résultat de l'analyse du flux vidéo (pour la direction du regard, mouvement de la tête, etc.) et des actions effectuées sur les éléments de contrôles (clic de souris, sélection d'une entrée, bouton, etc.). L'intérêt d'un tel modèle réside dans l'identification de certains comportements du joueur dans le but de contrôler l'exécution du jeu. Ils ont appliqué leurs résultats de recherche à un environnement interactif de jeux éducatifs destinés à des enfants autistes. Il s'agit donc des jeux interactifs simples, mono-joueur et sans adversaire mais capables de s'adapter aux profil et comportement du joueur.

1.3. Remerciements

Je tiens à remercier en premier lieu Imad Saleh sans qui la journée *Intelligence Artificielle et Jeux* n'aurait pas eu lieu, et sans qui ce livre n'existerait pas. Je remercie

aussi Rémi Coulom, Philippe Mathieu, Jean Méhat, Jacques Pitrat, Cédric Sanza et Olivier Sigaud pour m'avoir aidé à relire les chapitres de ce livre. Je remercie enfin Hélène, Clémentine, Cyprien et Charles pour leur bonne humeur communicative.

1.4. Bibliographie

[BIL 03] BILLINGS D., BURCH N., DAVIDSON A., HOLTE R. C., SCHAEFFER J., SCHAUENBERG T., SZAFRON D., « Approximating Game-Theoretic Optimal Strategies for Full-scale Poker. », *IJCAI*, p. 661-668, 2003.

[BOU 01] BOUZY B., CAZENAVE T., « Computer Go : An AI-Oriented Survey », *Artificial Intelligence*, vol. 132, n° 1, p. 39-103, October 2001.

[BUR 04] BURO M., « Call for AI Research in RTS Games », *Proceedings of the AAAI-04 workshop on AI in games*, San Jose, p. 139-142, 2004.

[LAI 01] LAIRD J. E., VAN LENT M., « Human-Level AI's Killer Application : Interactive Computer Games. », *AI Magazine*, vol. 22, n° 2, p. 15-26, 2001.

[ROM 03] ROMEIN J. W., BAL H. E., « Solving Awari with Parallel Retrograde Analysis », *IEEE Computer*, vol. 36, n° 10, p. 26-33, October 2003.

[SCH 97] SCHAEFFER J., *One Jump Ahead : Challenging Human Supremacy at Checkers*, Springer-Verlag, New York, NY, 1997.

[SHE 04] SHEPPARD B., « Efficient control of selective simulations », *ICGA Journal*, vol. 27, n° 2, p. 67-80, June 2004.

[TES 95] TESAURO G., « Temporal Difference Learning and TD-Gammon. », *Commun. ACM*, vol. 38, n° 3, p. 58-68, 1995.