UNIVERSITÉ PARIS NANTERRE

MÉMOIRE DE MASTER 2 MIAGE

Intégration de comportements humains dans les systèmes autonomes

---

Auteur :
Ana GVIMRADZE

Tutrice :
Sonia GUEHIS

17 Juin, 2019

« Computers will overtake humans with AI at some point within the next 100 years. When that happens, we need to make sure the computers have goals aligned with ours. »

– Stephen Hawking

Remerciements

J'aimerais d'abord remercier tous les membres de ma famille qui même si physiquement sont très loin de moi, me soutiennent depuis le premier jour où je suis arrivé en France.

Je remercie également ma tutrice ainsi que tout le corps professorial du Master MIAGE, ces deux années ont été riches en expérience.

Je tiens aussi à remercier l'équipe CRM de LVMH - PCIS pour leur aide et soutien tout au long de mon stage.

Pour finir je remercie tous mes amis qui m'ont soutenu dans les moments les plus dures et notamment pendant la réalisation de ce mémoire...

Table des matières

Table des figures

Liste d'abréviations

BDI	**B**elief, **D**esire, **I**ntention
IA	**I**ntelligence **A**rtificielle
NDM	**N**aturalistic **D**ecision **M**aking
RPD	**R**ecognition - **P**rimed **D**ecision making
SMA	**S**ystème **M**ulti - **A**gent

Dédié à Herbie qui me manque énormément...

Chapitre 1

Introduction

Dans cette partie, je donne une brève description du sujet de mémoire que j'ai développé tout au long de cette année.

Mon sujet s'intitule : Intégration de comportements humains dans les systèmes autonomes.

Je m'explique, nous connaissons en notre temps un très fort essor de nouvelles technologies de tous genres, celles-ci ont des buts divers et variés, que ce soit pour nous aider à choisir une playlist musicale afin de bien démarrer une journée de travail, ou encore pour nous aider à mieux gérer notre entreprise avec des logiciels dédiés, jusqu'aux applications les plus compliquées dans le domaine médical tel que l'assistance d'un chirurgien lors d'une opération risquée.

Parmi les technologies les plus avancées et les plus prometteuses on trouve les intelligences artificielles, ces machines auxquelles on apprend à apprendre.

Celles-ci sont capable de mémoriser une quantité de données très élevée afin de cerner leurs environnements et ainsi pouvoir agir efficacement dans le but d'aider les hommes.

On les trouve partout, au supermarché, à la banque, sur internet, dans les jeux vidéo etc.

Malheureusement, malgré toute leur puissance de calcul, celles-ci ne sont pas capables de comprendre et d'interpréter une chose élémentaire qui fait le propre de l'homme : ses sentiments.

En effet, les IA actuelles peuvent résoudre des problèmes très complexes, mais sans savoir ce qu'engendrent leurs prises de décisions sur le ressenti de l'homme et sa moralité, pourront-elles vraiment nous être bénéfiques ? Ou au contraire : destructrices.

C'est la question à laquelle j'essaye de répondre lors de ce mémoire en appliquant ce principe aux intelligences artificielles que l'on peut trouver dans les jeux vidéo. Essayer de leur inculquer non seulement ce qui les entoure, mais aussi ce qu'implique de prendre telle ou telle décision et ses répercussions sur l'homme, ou en l'occurrence dans mon cas, le joueur.

L'objectif est donc d'utiliser des androrithmes (LEONHARD, 2016) (algorithmes calqués sur le modèle humain et nourris par ce dernier) et de les appliquer à différentes intelligences artificielles dans des jeux vidéo open-source.

1.1 Prérequis

L'idée qu'un jour les robots puissent avoir des émotions a capturé l'imagination de beaucoup et a été dramatisée par des robots et des androïdes dans de nombreux films de science-fiction. Ce mémoire aborde la question de savoir si les robots peuvent prendre des décisions qui ne sont pas liées à un choix rationnel, ce qui supposerait que ceux-ci agissent sous le feu de l'émotion. Les études sur le cerveau et notamment sur les émotions reposent principalement sur l'étude des aspects sociaux, communicatifs, adaptatifs, régulateurs et expérientiels de celles-ci, ainsi la neurochimie révèle la manière dont différents «neuromodulateurs» tels que la sérotonine, la dopamine et les opioïdes peuvent affecter l'équilibre émotionnel du cerveau et les études de différentes régions telles que l'amygdale et le cortex orbifrontal fournissent une vue du cerveau comme un réseau de sous-systèmes en interaction. (ARBIB, 2005)

1.1.1 L'amygdale

L'amygdale (latin, corpus amygdaloideae) [Figure 1.1, Page 3] est un ensemble de neurones en forme d'amande situés au plus profond du lobe temporal médial du cerveau. L'amygdale, qui joue un rôle clé dans le processus des émotions, fait partie du système limbique. Chez les humains et les autres animaux, cette structure cérébrale sous-corticale est liée à la fois aux réactions de peur et au plaisir. Sa taille est en corrélation positive avec le comportement agressif d'une espèce à l'autre. Tels que l'anxiété, l'autisme, la dépression, le trouble de stress post-traumatique et les phobies sont soupçonnés d'être liés au fonctionnement anormal de l'amygdale, en raison de dommages, de problèmes de développement ou d'un déséquilibre neurotransmetteur.

1.1.2 Le cortex orbitofrontal (OF)

Le cortex orbifrontal [Figure 1.2, Page 3] est une région du cortex cérébral qui entre en jeu dans le processus de décision. Il est situé en position antérieure et sur la face inférieure du cortex préfrontal. Il prend son nom des lobes frontaux et du fait qu'il est situé au-dessus des orbites.

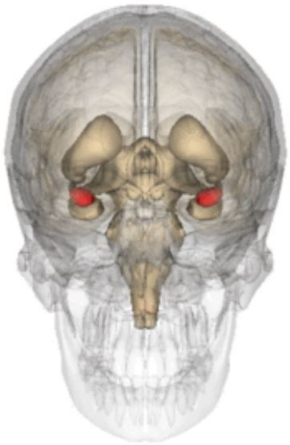

FIGURE 1.1 – Vue 3D de l'amygdale (en rouge).

Cette partie du cortex préfrontal est en connexion avec le thalamus. Parce qu'il est actif dans les émotions et le système de récompense, le cortex orbitofrontal est souvent considéré comme faisant partie du système limbique.(*Orbitofrontal Cortex*)

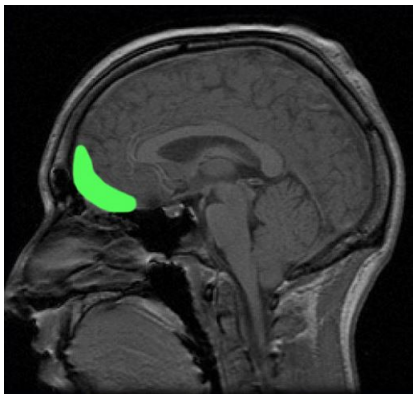

FIGURE 1.2 – Emplacement approximatif de cortex orbitofrontal.

1.1.3 Émotions

Chez les êtres humains, l'émotion inclut fondamentalement « un comportement physiologique, des comportements expressifs et une conscience » (MYERS, 2004)

Il existe certaines catégories des émotions :
— émotions « cognitives » par opposition aux émotions « non cognitives »
— émotions instinctives (des amygdales), par opposition aux émotions cognitives (du cortex orbitofrontal)
— émotions primaires et secondaires

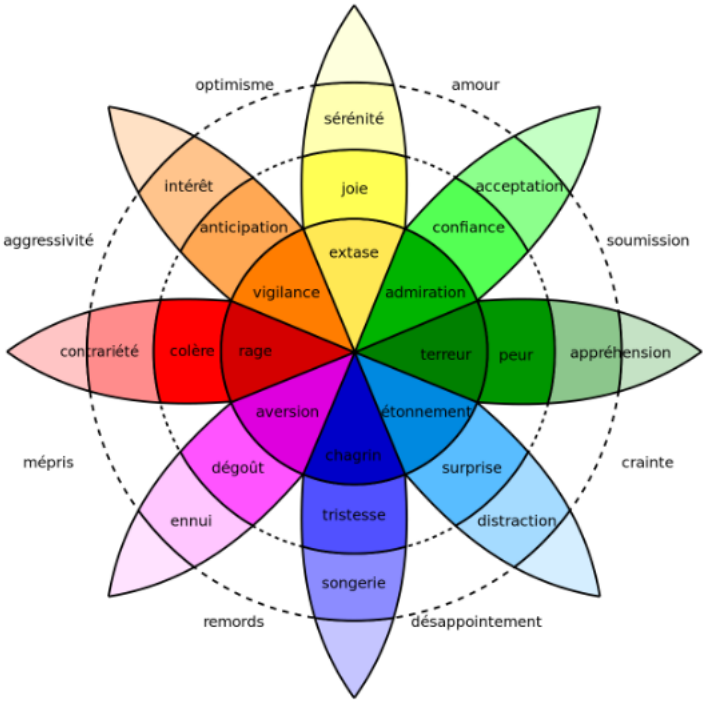

FIGURE 1.3 – Roue des émotions de Robert Plutchik

"La théorie psycho-évolutionniste des émotions de Robert Plutchik (*Robert Plutchik*) est l'une des méthodes de classification des réactions émotives générales. Plutchik considérait qu'il y avait huit émotions de base [Figure 1.3] : la joie, la peur, le dégoût, la colère, la tristesse, la surprise, la confiance et l'anticipation. Un professeur et psychologue américain Robert Plutchik propose ses 4 émotions fondamentales primaires (la peur, la colère, la joie, la tristesse), qui s'associent à des mécanismes de mémoire et de réflexion pour donner 4 autres émotions fondamentales secondaires (la confiance (liée à la joie), le dégoût

(lié à la tristesse), l'anticipation (liée à la colère) et la surprise (liée à la peur)), dont les fonctions respectives seraient la préservation, la protection des acquis, la reproduction, la réintégration, l'incorporation, le rejet, l'orientation et l'exploration). D'autres systèmes de classement des émotions, comme celui de Paul Ekman, ne considèrent que 4 à 6 émotions primaires au lieu de 8, car dans la mesure où les émotions combinées font intervenir des mécanismes de réflexion et de mémoire (par exemple la confiance est liée à un ensemble de souvenirs joyeux) voire de pensée abstraite, il ne s'agit plus d'émotions mais de sentiments par définition.

Plutchik a organisé ses émotions en paires d'opposés : la joie et la tristesse, la peur et la colère, le dégoût et la confiance, la surprise et l'anticipation. Il faut bien garder à l'esprit que chacune de ces émotions peut varier en intensité, ce qui génère un grand nombre de mots pour les décrire. Par conséquent, il y combine l'idée d'un cercle des émotions et celle d'une palette de couleurs pour représenter cette variation d'intensité et la classer en niveaux, même si la séparation entre les niveau n'est pas franche. Comme ces dernières, les émotions de base peuvent s'exprimer à divers degrés d'intensité et se combiner l'une à l'autre pour former des émotions différentes. C'est ainsi que Plutchik est venu à définir les dyades primaires [Figure 1.4, Page 6] (combinaisons de deux émotions de base adjacentes), secondaires (combinaisons d'émotions de base voisines à une émotion près) et tertiaires (combinaisons d'émotions de base voisines à deux émotions près) qui suivent (TAYARI, LE THANH et AMAR, 2009) : "

Dyades primaires	Résultats	Dyades secondaires	Résultats	Dyades tertiaires	Résultats
Joie et confiance	Amour	Joie et peur	Culpabilité	Joie et surprise	Ravissement[4]
Confiance et peur	Soumission	Confiance et surprise	Curiosité	Confiance et tristesse	Fadeur
Peur et surprise	Crainte	Peur et tristesse	Désespoir	Peur et dégoût	Honte
Surprise et tristesse	Désappointement	Surprise et dégoût	Horreur	Surprise et colère	Indignation[4]
Tristesse et dégoût	Remords	Tristesse et colère	Envie	Tristesse et anticipation	Pessimisme
Dégoût et colère	Mépris	Dégoût et anticipation	Cynisme	Dégoût et joie	Morbidité
Colère et anticipation	Agressivité	Colère et joie	Fierté	Colère et confiance	Domination
Anticipation et joie	Optimisme	Anticipation et confiance	Fatalisme	Anticipation et peur	Anxiété

FIGURE 1.4 – Dyades primaires, dyades secondaires, dyades tertiaires et leurs résultats selon la théorie de Plutchik

Chapitre 2

Ordinateurs et émotions

2.1 Objectifs du mémoire

Ce mémoire a pour objectif de contribuer à rendre les agents autonomes plus crédibles aux yeux des êtres humains qui interagissent avec eux, en effet, à l'heure actuelle, les agents autonomes que l'on peut retrouver dans les différentes simulations de manière opérationnelle remplissent très bien leurs tâches quand il s'agit de tests d'équipements, de nouvelles tactiques, ou encore de traitement de grandes quantités de données visant à la prise de décision, ce qui permet aux humains d'analyser les données résultantes de ces simulations et de les appliquer sur le terrain.

Le problème qui se pose dans l'état actuel, c'est que lorsque l'être humain est mis dans la boucle de ces simulations et interagit de manière directe avec ces agents, ceux-ci n'ont aucune crédibilité à ses yeux quant à la manière dont ils prennent telle ou telle décision ou décident d'appliquer telle ou telle tactique, l'échange entre humain et machine peut vite devenir incompréhensible et aboutir à des résultats inattendus.

En effet les agents actuels procèdent de manière rationnelle pour faire un choix parmi une multitude de possibilités, ce choix est celui-ci ayant acquis le plus de "points" où "score le plus élevé" durant les différentes phases précédentes il est mis au-dessus de la pile et est considéré comme "le choix de prédilection".

Or l'être humain n'utilise que très rarement ce type de prise de décision, les NDM (natural decision making) ou prise de décision dans un milieu naturel prouvent que les choix rationnels ne s'appliquent pas aux paramètres du monde réel en raison d'un certain nombre de facteurs qui rendent son application impossible ou difficile.

Contribuer à rendre plus réaliste la prise de décision de systèmes autonomes dans un milieu naturel (pas que les choix rationnels)(à développer).-> proposer un agent autonome de jeu vidéo auquel un modèle de prise de décision dans un environnement naturel (NDM) est appliqué et voir les résultats de cette application (fonctionnement/non fonctionnement).

2.1.1 Objectif concret (framework)

La solution que je propose est un framework développé C sur le logiciel Unity, logiciel de création de jeux vidéo, cette solution modélise un agent autonome auquel une succession de paramètres de prises de décisions en milieu naturel est appliquée, il sera ensuite confronté à des facteurs et situations inspirées du monde réel devant lesquels il devra faire des choix “naturels”.

Cette solution a une valeur ajoutée par rapport à l’existant, en effet, dans le monde du jeu vidéo, la remarque la plus fréquente des joueurs est que les agents auxquels ils se retrouvent confrontés lors de leurs sessions de jeu sont très loin d’atteindre le réalisme des joueurs humains, et cela même si la difficulté sélectionnée est la plus élevée.

En effet, le paramètre de difficulté ne fait que rendre l’adversaire plus rapide, plus endurant et moins sensible aux dommages corporels, mais ne lui acquiert en aucun cas un comportement plus humain, de peur, de courage ou encore de stratégie, ce qui rend l’expérience de jeu moins crédible et moins intéressante, cela explique aussi le succès phénoménal des jeux en ligne, où dans ce cas, le joueur est confronté à un autre être humain, qu’il peut comprendre, déterminer ses mouvements et ses choix selon les situations et la configuration de l’instant, une “intelligence artificielle” sophistiquée représente donc un facteur marketing clé pour les entreprises de jeux vidéo.

Chapitre 3

Background

3.1 SMA (ou Système Multi-agent)

En informatique, un système multi-agent (SMA) est un système composé d'un ensemble d'agents (un processus, un robot, un être humain, etc.), situés dans un certain environnement et interagissant selon certaines relations. Un agent est une entité caractérisée par le fait qu'elle est, au moins partiellement, autonome. Objet de longue date de recherches en intelligence artificielle distribuée, les systèmes multi-agents forment un type intéressant de modélisation de sociétés, et ont à ce titre des champs d'applications larges, allant des sciences humaines, jusqu'aux services militaires et médicaux. (*SMA*)

3.1.1 Exemple de SMA de jeu vidéo

Les communautés virtuelles que l'on trouve de plus en plus dans les jeux vidéo sont un parfait exemple afin de mieux comprendre le fonctionnement d'un SMA, par exemple, un jeu qui simulerait la vie d'une famille, plusieurs dimensions composent alors ce SMA.

Premièrement, un environnement ayant comme paramètre sa taille qui pourrait se caractériser par la maison et son jardin. Deuxièmement, les agents de ce SMA disposent d'une quantité d'objets dits "passifs" avec lesquels ils peuvent interagir, ce sera l'équipement de la maison ou encore la nourriture. Ensuite, les agents eux-mêmes, actifs et autonomes, ils sont en contact avec tout ce qui les entoure, leurs environnements, les objets qui le composent ou encore les autres agents, on les identifie comme étant "les membres de la famille". On intègre ensuite le concept d'organisation, constitué des différentes relations entre les objets et les agents, liens familiaux, notions de propriété(qui possède tel ou tel objet). Pour finir, on ajoute des opérateurs qui permettent aux agents d'agir sur leur environnement ou sur les autres agents (le fils peut manger un yaourt, promener

son chien ou parler à sa sœur) Enfin, on intègre un ensemble d'opérateurs qui permettent aux agents d'agir sur les objets ou sur les autres agents (le fils peut promener son chien ou manger un yaourt ou parler à son père), et de capteurs qui permettent aux agents de connaître les changements de l'environnement et des autres agents (le yaourt est tombé par terre, papa m'a demandé de sortir le chien). Voici donc ce que l'on peut appeler un SMA. (*SMA*)

3.1.2 Exemple de SMA en opération "SWARMM" (Smart Whole Air Mission Model)

le programme "SWARMM" a été développé par la division des opérations aériennes de l'organisation australienne de défense de science et technologies, il a pour objectif de simuler les opérations des avions de combat pour la flotte aérienne australienne "La Royal Australian Air Force".

Chaque pilote est un agent programmé avec dMARS, un environnement de programmation basé sur BDI(Belief-desire-intention ou Croyance-désir-intention) implémenté en FORTRAN et en C, chacun d'entre eux recevant des données des modèles physiques équivalents à ceux qu'un pilote reçoit de sa vision et de ses instruments et effectuant ce qui suit :

— Cycle de prise de conscience de la situation lorsque les données sont plus complexes (descripteurs symboliques)
— Évaluation de la situation, la situation est basée sur les résultats précédents de l'étape précédente.
— Sélection tactique, sélectionnée en fonction de l'ensemble actuel d'objectifs qui a été reconnu lors des étapes précédentes
— Procédures d'opération : choisies pour mettre en œuvre une tactique.

Plus d'une décennie de contacts étroits, d'entretiens avec des combattants, de briefing de missions et d'implication dans des exercices d'entraînement a été nécessaire afin d'arriver à établir ce cycle, il s'est avéré être d'une très grande valeur car il permet une intégration simple des procédures standard et de manière très documentée, il permet aussi aux combattants de décrire facilement leurs connaissances en se basant sur les différentes étapes du cycle et pour finir, il assure un échange simple compris entre les deux partis, pilotes et programmeurs, les programmeurs peuvent ainsi décrire leur résultat et débattre avec les pilotes en utilisant des termes

similaires, il est principalement basé sur le travail d'équipe et s'est avéré être extrêmement utile pour tester de nouveaux équipements et tactiques.

L'un des principaux buts du projet était l'introduction d'un pilote (agent humain) dans la boucle de simulation en tant que coéquipier ou adversaire, il a donc été prouvé que même si pour un observateur extérieur ce programme ressemble à un vrai pilote, il n'était pas crédible aux yeux des vrais pilotes, et cela aux vues de son comportement extrêmement rationnel et peu naturel. (NORLING, SONENBERG et RÖNNQUIST, 2000)

3.2 NDM (Naturalistic Decision Making)

> " L'étude de la NDM pose la question de comment des individus expérimentés, travaillant individuellement ou en groupes, dans un environnement dynamique, rapide et incertain, identifient et évaluent leurs situations, prennent des décisions et effectuent des actions qui ont des conséquences signifiantes sur eux ainsi que sur l'organisation dans laquelle ils opèrent " (ZSAMBOK, 2014)

Ce terme est apparu en 1989 lors d'ateliers de chercheurs qui avaient pour but d'étudier "les prises de décisions dans des contextes réalistes" (médecine, centrales nucléaires, planification exécutive), leurs études ont montrés que les théories classiques sur les prises de décisions n'étaient tout simplement pas applicables dans le monde réel, ils ont aussi prouvé que même lorsque les agents étaient entraînés à faire des choix rationnels, ces derniers ne le faisaient que rarement.

Il en a ainsi émergé une meilleure compréhension sur la manière dont nous procédons pour prendre des décisions dans des situations complexes, dans certains cas, nous procédons effectivement de manière rationnelle, où, une multitude d'options sont générées, et la "meilleure" est sélectionnée, mais il existe d'autres stratégies qui sont plus communément utilisées.

La recherche dans ce domaine est principalement destinée à la conception d'aides à la décision, mais ces résultats peuvent également être utilisés pour développer de meilleurs modèles cognitifs pour les humains lors des simulations.

Orasanu et Connolly (ORASANU et CONNOLLY, 1993) énumèrent huit facteurs qui caractérisent les paramètres de prises de décisions en milieu

naturel. De nombreuses études de prise de décision classiques ignorent ou limitent délibérément ces facteurs, ce qui rend la théorie du choix rationnel plus facile à appliquer.

Ces facteurs sont :
— Problèmes mal structurés.
— Environnements dynamiques incertains.
— Objectifs changeants, mal définis ou concurrents.
— Boucles d'action / de rétroaction.
— Le stress dû au temps.
— Des enjeux trop élevés.
— Plusieurs joueurs.
— Objectifs organisationnels et normes.

Tous les facteurs ne sont pas présents dans un contexte dit "naturel", mais chacun ajoute une complexité au problème.

3.2.1 Modèles de NDM

Plusieurs modèles de prise de décision naturaliste ont été proposés, mais à ce jour, aucun d'entre eux ne rend compte du spectre complet des décisions pouvant être prises dans un contexte naturel. Lipshitz (LIPSHITZ, 1993) donne un résumé de neuf modèles qu'il souligne entre non contradictoires, mais illustrent les différents types de prise de décision qui peuvent être utilisés.

Notez que les chercheurs dans ce domaine s'intéressent généralement aux personnes qui sont expérimentées dans leur domaine. Hubert Dreyfus explique «le modèle d'acquisition de compétences en cinq étapes dans (DREYFUS, 2014), avec des niveaux allant de novice (quelqu'un qui débute dans le domaine, comme un apprenti pilote), à un expert (quelqu'un qui est très habile et immergé dans son activité). La plupart des modèles NDM supposent un certain niveau d'expertise dans le domaine, pas nécessairement expert, mais certainement pas novice. L'un des modèles les plus connus est celui de Klein intitulé "recognition-primed decision making" Modèle (RPD), ou "la prise de décision axée sur la reconnaissance". (KLEIN, 2017), illustré à la Figure 3.1.

Comme le souligne Klein lui-même «Le modèle RPD n'est pas issue de recherches en NDM »((KLEIN, 2017), p.102), mais des études d'experts dans divers domaines montrent qu'une grande partie de leurs décisions

sont prises de cette façon (voir le tableau 7.2 dans (KLEIN, 2017)). La chose importante à noter à propos de ce modèle est l'accent qui est mis sur l'évaluation de la situation. Une fois que le décideur a reconnu la situation, il existe quatre sous-produits :

1. Il / elle s'attend à ce que certaines choses se produisent mais pas d'autres.
2. Il / elle fait attention à certains signaux pour soutenir le diagnostic.
3. Il existe une certaine compréhension des objectifs qu'il est plausible de réaliser.
4. et certaines actions sont susceptibles de réussir.

Le décideur choisit ensuite une action, exécute une rapide simulation mentale de celle-ci, et s'il/elle pense que cela va réussir, la met en œuvre. Une fois que le plan d'action a été sélectionné et qu'il est entamé, la situation est surveillée pour s'assurer qu'elle se déroule comme prévu, sinon, d'autres actions pourraient être envisagées, mis à part cette éventualité, le décideur n'envisage pas d'autres options. (Notez que cet aspect du modèle n'est pas indiqué dans le diagramme.) Dans le modèle RPD, un opérateur expérimenté choisira généralement «automatiquement» un plan d'action une fois que la situation est reconnue, et que ce plan d'action est susceptible d'être celui qui a donné auparavant des résultats positifs dans la même situation.

Ce modèle [Figure 3.1, Page 14] exprime l'idée qu'une personne qui acquiert de l'expertise est plus susceptible de reconnaître les subtilités entre les situations et de choisir un plan d'action en conséquence.

3.3 Une brève histoire des jeux dans la recherche sur l'IA

Les jeux ont une longue histoire dans la recherche sur l'IA, remontant au moins à 1949 lorsque Claude Shannon (peu après avoir développé l'entropie d'informations) s'est intéressé à l'écriture d'un programme informatique pour jouer au jeu d'échecs. Dans son article «Programmer un ordinateur pour jouer aux échecs», Shannon écrit :

"La machine à échecs est un système idéal pour commencer, car :

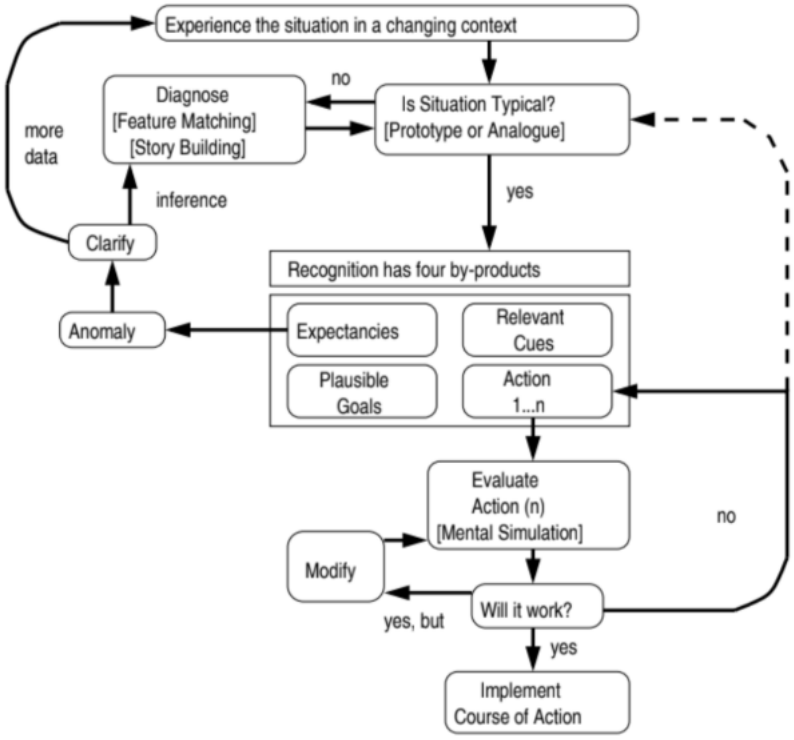

FIGURE 3.1 – La version intégrée du modèle de Klein de décision axée sur la reconnaissance (Figure 7.1 du livre "Sources of Power" by G. Klein 1998 (KLEIN, 2017))

1. Le problème est clairement défini à la fois dans les opérations autorisées (les mouvements) et dans le but ultime (le maté) ;
2. Il n'est ni si simple d'être trivial ni trop difficile pour une solution satisfaisante ;
3. Les échecs sont généralement considérés comme nécessitant une «réflexion» pour un jeu habile ; une solution de ce problème nous obligera soit à admettre la possibilité d'une pensée mécanisée, soit à restreindre davantage notre concept de «pensée» ;
4. La structure discrète des échecs s'intègre parfaitement dans la nature numérique des ordinateurs modernes."

C'était en 1949.

Depuis lors, il existe un intérêt durable pour la création de programmes informatiques capables de jouer à des jeux aussi habiles que des joueurs

humains, même en battant les champions du monde respectifs. Shannon a inspiré le travail fondateur d'Arthur Samuel sur Checkers dans les années 1950 et 1960. Si le programme de Samuel n'a pas pu battre les experts, il a été considéré comme une réalisation majeure, car il s'agissait du premier programme à utiliser efficacement les procédures de recherche heuristique et les méthodes basées sur l'apprentissage. (*Introducing : Unity Machine Learning Agents Toolkit*, 2017)

Chinook, un programme de contrôleurs mis au point à l'Université de l'Alberta en 1989, a commencé à battre les joueurs les plus humains et, dès 1994, les meilleurs joueurs pouvaient au mieux égaliser la machine. Cette tendance s'est poursuivie avec d'autres jeux de plateau à 2 joueurs tels que Backgammon (avec TD-Gammon de Gerald Tesauro, 1992-2002) et Chess (lorsque Deep Blue d'IBM a battu Garry Kasparov, 1997), et plus récemment avec Go.

Une avancée scientifique importante de ces dernières années a été celle où, en 2016, AlphaGo de DeepMind a battu le champion du monde 18 fois Lee Sedol 4 à 1, faisant l'objet du documentaire Netflix, AlphaGo.

Chapitre 4

BDI (Belief-Desire-Intention)

Les agents de BDI sont basés sur les concepts philosophiques d'intentions, de plans et de raisonnements pratiques développés par Bratman (BRATMAN, 1987). Ce modèle est basé sur la psychologie populaire, c'est-à-dire la manière dont nous pensons. Le modèle fournit une première approximation de la cognition humaine, mais il reste encore beaucoup à faire pour l'affiner.

4.1 Belief

Les croyances d'un agent sont sa vision du monde, qui n'est pas nécessairement la même chose que l'état du monde, car les capteurs peuvent être imparfaits, en effet les informations fournies peuvent être à la fois incomplètes et bruyantes.

4.2 Desire

Plutôt que les désirs d'un agent, nous nous référons à ses objectifs. Ceux-ci donnent l'état du monde dans lequel l'agent souhaite être et doit être cohérent.

4.3 Intention

Ses intentions sont les plans qu'il exécute actuellement. Il peut y avoir plus d'un plan en cours, car un agent peut travailler simultanément à plusieurs objectifs (non conflictuels). Une fois qu'un agent a formulé une intention (c.-à-d. Qu'il sélectionne un plan), il est en quelque sorte engagé dans ce plan - il continue de l'exécuter (ou du moins a l'intention de l'exécuter) jusqu'à ce que l'objectif soit atteint, ou qu'il devienne impossible à atteindre en suivant ce même plan, l'objectif devient alors inutile. Un plan

est une «recette» pour atteindre un objectif particulier. C'est une séquence d'actions et / ou de sous objectifs à réaliser. Si une étape de la séquence échoue, le plan lui-même échouera. L'une des caractéristiques d'un système BDI est que, lorsqu'un plan échoue, l'agent réessaye (si possible). Il tentera de trouver un autre moyen d'atteindre l'objectif en tenant compte du fait que le monde (et donc les convictions de l'agent ou son Desire) est en train de changer. Un agent stocke ses plans dans une bibliothèque de plans.

4.4 Agent BDI

L'agent montré à la Figure 4.1, Page 19 passe par un cycle continu de :

1. Ressentir l'environnement.
2. Raisonner sur les croyances, les objectifs et les intentions.
3. Accomplir une ou plusieurs actions.

Ce cycle est très similaire à celui utilisé dans SWARMM section 3.1.2, qui sépare la deuxième étape en deux étapes : évaluation de la situation suivie de la sélection tactique. En effet, SWARMM est implémenté en utilisant une architecture BDI. Au cours de la phase de raisonnement du cycle, l'agent doit raisonner sur les croyances (si et comment elles devraient changer), les objectifs (les changements de croyances peuvent affecter la faisabilité des objectifs) et les intentions (les changements d'objectifs peuvent amener l'agent à abandonner certaines intentions et / ou d'en créer de nouvelles). L'agent doit également décider quelle action (ou quelles actions) effectuer ensuite, à partir des intentions actuelles.

Lorsqu'il existe plusieurs plans disponibles pour atteindre un objectif donné, l'agent utilise en théorie un choix rationnel pour sélectionner un plan (BRATMAN, ISRAEL et POLLACK, 1988). C'est-à-dire que les avantages de tous les plans applicables sont évalués et que le «meilleur» est sélectionné. Cependant la recherche en NDM indique que ce n'est pas ainsi que les gens prennent leurs décisions, et c'est sur cela que le travail doit être fait pour améliorer le modèle BDI .

Dans les implémentations pratiques d'architectures BDI, telles que JACK (*Agent Oriented Software JACK*) ou dMARS (D'INVERNO et al., 1997), chaque plan est conçu pour gérer un objectif particulier dans un contexte particulier. (Dans ces systèmes, le "contexte" est un ensemble de conditions que l'agent doit croire vraies.) Cela permet au programmeur de spécifier différentes manières d'atteindre le même objectif dans différentes situations,

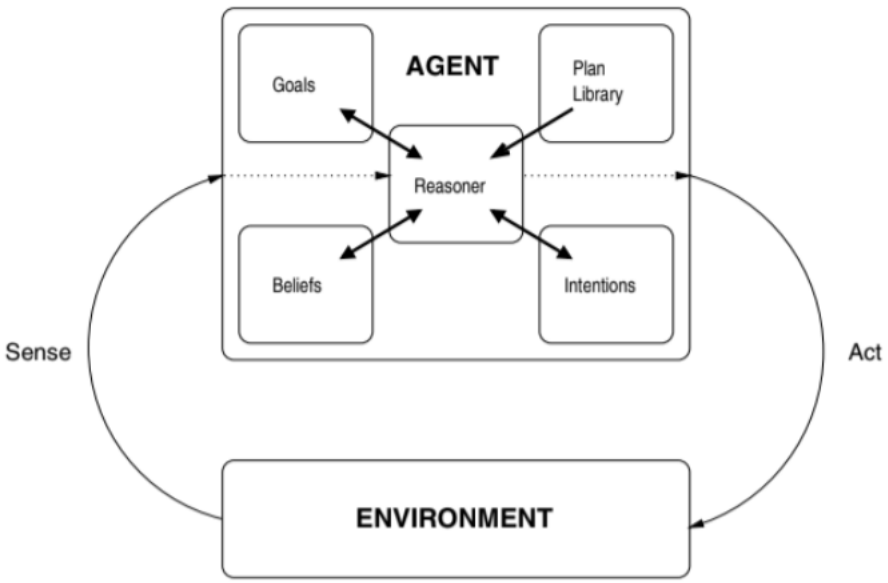

FIGURE 4.1 – Structure d'un agent BDI

mais il est également possible d'avoir plusieurs plans applicables dans une situation donnée (lorsque les contextes se chevauchent).

Chapitre 5

Autres modèles et architectures

5.1 CogAff

CogAff est un modèle de traitement de l'information proposé par Sloman (SLOMAN, CHRISLEY et SCHEUTZ, 2005). Il comprend trois niveaux, à savoir : réactif, délibératif et réflexif (méta-gestion)). Chaque niveau comprend des mécanismes de perception, des mécanismes de traitement centralisés et des mécanismes d'action. L'architecture possède un mécanisme d'alarme ainsi que des communications d'informations entre toutes les parties de l'architecture. Le mécanisme d'alarme fonctionne de manière centralisée et s'apparente au mécanisme d'interruption.

5.1.1 H-CogAff

H-CogAff est un exemple particulier de CogAff expliquant les phénomènes mentaux humains. Chaque niveau supporte différentes catégories d'émotions. «D'autres subdivisions sont nécessaires pour couvrir toute la variété des émotions humaines, d'autant plus que les émotions peuvent changer de caractère au fil du temps, à mesure qu'elles grandissent» (SLOMAN, CHRISLEY et SCHEUTZ, 2005).

5.2 CLARION

CLARION a une structure similaire aux trois niveaux de traitement de l'information. C'est une architecture cognitive qui comporte deux niveaux : le niveau implicite (similaire aux niveaux réactifs et de routine) et le niveau explicite (similaire au niveau réflexif). Le niveau implicite est le niveau inférieur et code les connaissances implicites. Il utilise un réseau de neurones multicouches avec Q-learning pour acquérir des connaissances

implicites. Le niveau explicite est le niveau supérieur et code la connaissance explicite. Il utilise un apprentissage ponctuel pour acquérir des connaissances explicites.

Le niveau le plus bas et le niveau le plus élevé échangent leurs apprentissages au moyen d'un apprentissage ascendant et descendant. Chaque niveau comprend quatre sous-systèmes fonctionnels distincts :

1. Un sous-système centré sur l'action pour contrôler les actions.
2. Un sous-système non centré sur l'action pour conserver les connaissances générales implicites ou explicites.
3. Un sous-système métacognitif pour surveiller, diriger et modifier les opérations de tous les sous-systèmes.
4. Un sous-système de motivation pour fournir les motivations sous-jacentes de la perception, de l'action et de la cognition, en termes d'impulsion et de rétroaction.

5.3 SHAME

SHAME (Architecture évolutive et hybride pour le mimétisme des émotions) est un modèle émotionnel ou système simulant l'état émotionnel d'un agent agissant dans un environnement virtuel présenté par Kesteren en 2001 (KESTEREN, 2001). Il a construit un modèle de simulation à base d'agents appelé «GridWorld». SHAME implémente la fonction d'affect similaire à la fonction d'affect dans le niveau de réflexion et des fonctions de motivation et de comportement similaires à celles du niveau de réactif. Il utilise un réseau de neurones pour apprendre comment l'état émotionnel devrait être influencé par la survenue de stimuli.

5.4 Zamin

Zamin est un environnement de simulation de vie artificielle pour évaluer les capacités des agents à produire un comportement émotionnel et à prendre de meilleures décisions, développé par Zadeh, Shouraki et Halavati (2006) (ZADEH, SHOURAKI et HALAVATI, 2006) ces derniers ont mis en place cet environnement afin d'étudier le possible rôle des émotions dans la gestion des ressources mentale. Ils utilisent uniquement un comportement positif/négatif et un comportement d'approche/d'évitement (uniquement la fonctionnalité du niveau réactif) dans un environnement prédateur-proie.

Chapitre 6

Contribution

6.1 Unity

Unity le logiciel que j'utilise pour développer mon framework, c'est est un moteur de jeu multiplateforme (smartphone, ordinateur, consoles de jeux vidéo et Web) développé par Unity Technologies. Il est l'un des plus répandus dans l'industrie du jeu vidéo, aussi bien pour les grands studios que pour les indépendants du fait de sa rapidité aux prototypages et qu'il permet de sortir des jeux sur tous les supports.

En terme d'intelligence artificielle, ce dernier propose une multitude de fonctionnalités notamment basées sur l'apprentissage (Machine Learning) afin de construire des agents autonomes, conscients de leur environnement et qui interagissent avec selon des paramètres précis, objectifs, actions et réactions.

L'un des facteurs qui m'ont poussé à l'utiliser et aussi le fait qu'il a la particularité de proposer une licence gratuite dite « Personal » avec quelques limitations de technologie avancée au niveau de l'éditeur, mais sans limitation au niveau du moteur.

6.2 Le Jeu open source "Do not shoot Aliens" - Jeu Mobile

L'idée principale derrière est de complètement transformer les mécanismes que l'on attend d'un jeu de tir classique, en effet, au lieu de viser et de tirer afin de vaincre ses ennemis, le personnage du joueur est un peu agressif et décide de tirer sur tout ce qui est autour de lui.

Le but du joueur est d'atteindre le moins de personnages extraterrestres pour ne pas les contrarier.

Plus les extraterrestres sont touchés, plus le jeu sera difficile, car le joueur devra les frapper une seconde fois pour les arrêter. [Figure 6.1]

La victoire est basée sur le fait d'accumuler le plus de points possible en allant de checkpoint en checkpoint.

FIGURE 6.1 – Icon du jeu "Do Not Shoot Aliens"

6.3 Langage de programmation

Chaque agent de ce jeu, que ce soit le personnage principal contrôlé par le joueur ou les agents autonomes représentés ici par les ennemis et même l'environnement ou les objets qui interagissent avec les agents, est géré par un code source qui lui est propre, ce dernier est écrit en C-Sharp, un langage de script qui permet ensuite au moteur de jeu de le compiler et de l'interpréter, et cela, afin de garantir les meilleures performances de jeu.

Ce langage est orienté objet, ce qui fait sens car tous les objets, environnements, agents ou encore joueurs sont effectivement des objets 3D, ayant des paramètres qui leur sont propres (Vitesse, durée de vie, couleur, objectif) ainsi que de nombreuses fonctions (marcher, courir, frapper, sauter etc...) qui leur permettent d'effectuer des actions, cette définition nous

renvoie au concept de classe, variables de classes et méthodes de classe présentes dans les langages orientés objet.

6.4 Description des agents autonomes présents dans le jeu

Le scénario typique pour la formation d'agents dans des environnements virtuels consiste à avoir un environnement et un agent uniques qui sont étroitement couplés. Les actions de l'agent modifient l'état de l'environnement et lui procurent des récompenses. (*Fostering AI Research : Meet us at AAAI-19*, 2019)

Dans notre cas, les agents ont pour environnement l'esplanade sur laquelle ils circulent, ils peuvent effectuer des actions par exemple : marcher doucement, courir ou encore attaquer.

Leur objectif initial est celui de rester "tranquille", en effet, tant qu'ils ne sont pas touchés, ils conservent leur démarche lente et continuent de circuler de manière hasardeuse (fonction random qui génère leur position dans l'espace).

Par contre s'ils venaient à être touchés par les balles du joueur principal, leur comportement change et leur objectif aussi, ils se concentrent alors principalement sur l'attaque du personnage principal afin de s'en débarrasser.

On peut identifier ici un exemple de BDI (décrit plus haut) :

— Leur "Belief" (ou croyance) initiale est de croire que leur environnement est inoffensif, c'est leur vision du monde.
— Ce qui engendre leur désire ou encore leur objectif, qui est de circuler dans l'environnement qui les entoure, ceux-ci décrivent l'état du monde parfait dans lequel ils aimeraient évoluer.
— Leur "Intention" est ici le plan d'action en cours d'exécution, c'est-à-dire, le fait de déambuler sur l'esplanade.

Comme vu précédemment, tout changement dans leur environnement peut produire un changement dans leur perception de ce dernier, et donc engendrer un nouveau plan d'action, c'est sur ce principe que je me base afin d'appliquer à ces gens différents paramètres qui leur permettront de réagir à leur nouvel environnement.

6.5 Application de paramètres de prises de décisions en milieu naturel (NDM) aux agents BDI

Afin d'appliquer ces paramètres, je procède de la manière suivante [Figure 6.2] :

— Je déclare des paramètres qui permettent d'évaluer l'état émotionnel des agents lors des différents changements de l'environnement.
— Ces variables sont : la peur, la joie, la tristesse et la colère qui sont des booléens.
— Chaqu'une de ces variable a, à son tour des paramètre qui lui sont étroitement liées : la vitesse de la démarche, ainsi que la couleur des agents.
— En l'occurrence, la couleur associée à la joie est le jaune.
— Celle associée à la tristesse est le bleue.
— Celle associée à la colère est le rouge.
— Celle associée à la peur est le vert. [Figure 6.3, Page 27]

```
// Les émotions primaires
bool spawned;
bool angry;
bool joy;
bool sad;
bool fear;
```

```
//Couleurs dépendants de l'émotion ( selon la roue des emotions de Robert Plutchik)
public Color normalColor;
public Color angryColor; //Rouge
public Color joyColor;  //Jaune
public Color sadColor; //Bleu
public Color fearColor; //Vert
```

```
//Variation de la vitesse de démarche dépendant de l'état émotionnel
public float walkspeed;
public float angryspeed;
public float joyspeed;
public float sadspeed;
public float fearspeed;
```

FIGURE 6.2 – Déclaration des variables

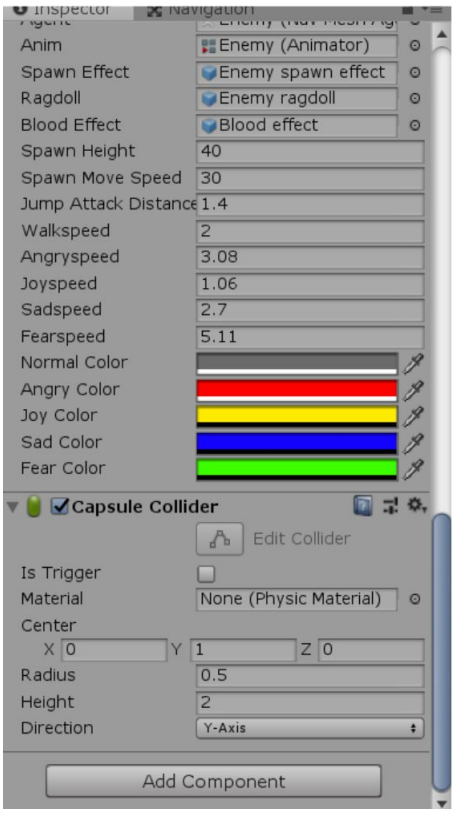

FIGURE 6.3 – Fenêtre Unity, Valeurs des paramètres

6.5.1 Première expérimentation

Lors de ma première expérimentation, j'ai décidé d'établir une distance critique entre le joueur et les agents autonomes, lorsque ce dernier est très proche d'eux, qui serait synonyme de tristesse pour ces derniers, sentiment lié au fait de savoir que le jour peut à tout moment les toucher.

La distance est exprimée ici par la position de l'agent en mode "stop" ou à l'arrêt (qui survient lorsque l'agent rencontre sur son chemin ou "path" le joueur) et le joueur lui-même, à laquelle on ajoute un delta en nombre flottant représenté ici par le 8f, il suffit ensuite de varier ce nombre flottant afin d'augmenter ou de diminuer le delta, et de calculer la distance qui sépare les agents autonomes(plus ou moins loin selon le delta) du joueur.

Le résultat souhaité est de pouvoir appliquer un changement de couleur aux agents suivant l'émotion qu'ils ressentent, dans la figure 6.4, c'est

```
// Si le tireurse se rapproche des cibles ,
//les cibles passent en état de peur (couleur rouge) et fuient (augmentent leur vitesse)
else if (Vector3.Distance(transform.position, player.position) > agent.stoppingDistance + 8f)
{
    rend.material.color = fearColor;
    agent.speed = fearspeed;
    RandomWalk();
}
```

FIGURE 6.4 – Code source implémentant une courte distance entre le jouer et l'agent

la peur, la couleur de l'agent prend donc la couleur associée à la peur [Figure 6.5] une fois que ce dernier est à une distance équivalente à un delta de 8, ce dernier change aussi de vitesse de démarche et adopte un rythme plus élevée lui permettant d'échapper au joueur au fur et à mesure que celui-ci se rapproche.(la vitesse liée à la peur est plus élevé que celle liée à "démarche dite normale comme le montre la figure-fenêtre-unity").

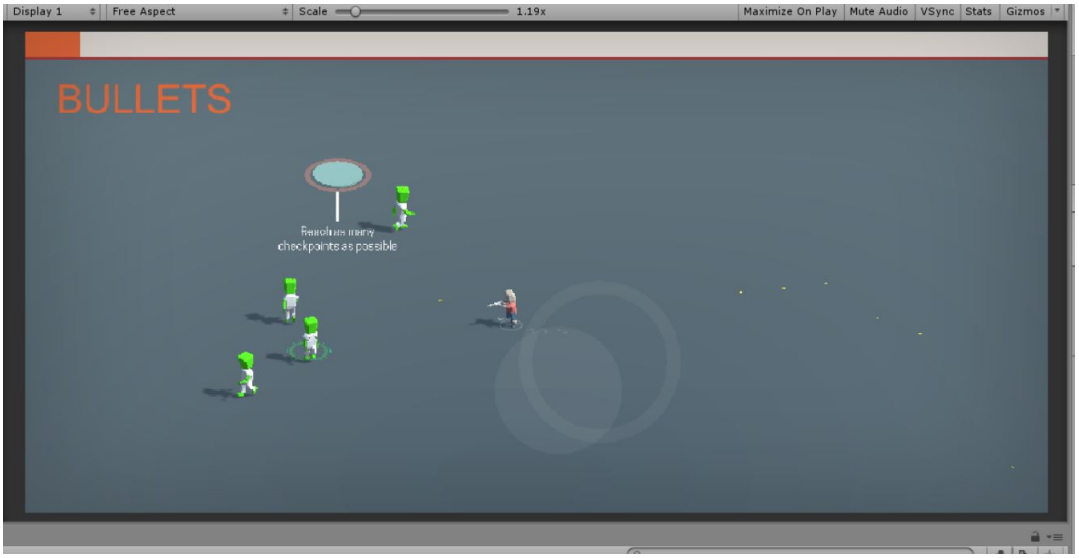

FIGURE 6.5 – Visualisation d'agents en état de peur

6.5.2 Deuxième expérimentation

On ajoute cette fois-ci à l'expérience précédente un nouveau facteur émotionnel, celui de la joie. Suivant le même principe que précédemment,

on définit un delta qui permet de calculer la distance entre le joueur et l'agent autonome, celui-ci est de 20f, une distance assez élevée. [Figure 6.6]

```
// Si la distance entre le tireur et les cibles est assez élévé
// - les cibles passent en état de joie ( couleur jaune ) et diminuent leur vitesse
else if (Vector3.Distance(transform.position, player.position) > agent.stoppingDistance +20f)
{
    rend.material.color = joyColor;
    agent.speed = joyspeed;
    RandomWalk();

}
```

FIGURE 6.6 – Code source implémentant une longue distance entre le jouer et l'agent

Le résultat souhaité est de pouvoir appliquer un changement de couleur aux agents à l'instant où cette distance entre eux et le joueur est atteints.

La couleur de ces derniers devient alors jaune, et leur vitesse passe à un rythme beaucoup moins élevée. [Figure 6.7]

FIGURE 6.7 – Visualisation d'agents en état de joie

6.5.3 Troisième expérimentation

On ajoute cette fois-ci à l'expérience précédente un autre facteur émotionnel, celui de la tristesse, ce dernier intervient lorsque la distance équivaut à la valeur médiane entre les deux distances précédentes, c'est-à-dire un delta de 14f. [Figure 6.8]

```
// Si le tireurse se rapproche des cibles ,
//les cibles passent en état de tristesse(couleur bleue) et fuient
else if (Vector3.Distance(transform.position, player.position) > agent.stoppingDistance + 14f)
{
    rend.material.color = sadColor;
    agent.speed = sadspeed;
    RandomWalk();
}
```

FIGURE 6.8 – Code source implémentant une distance médiane entre le jouer et l'agent

Le résultat souhaité est de pouvoir appliquer un changement de couleur aux agents à l'instant où cette valeur médiane est atteinte.

La couleur de ces derniers devient alors bleue, et leur vitesse passe à un rythme un peu plus soutenu que celui généré par la joie. [Figure 6.9]

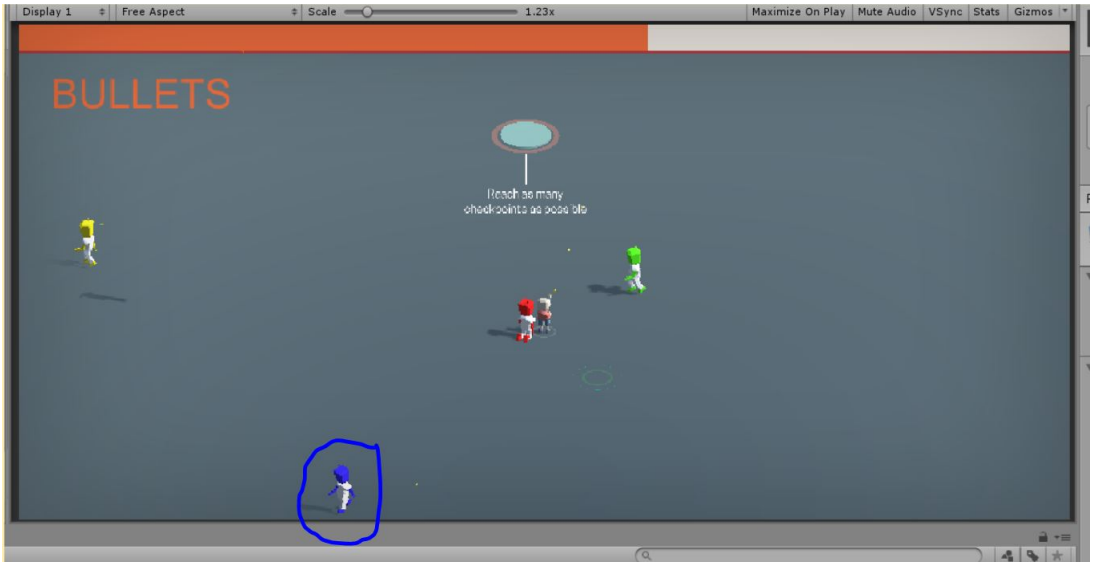

FIGURE 6.9 – Visualisation d'agents en état de tristesse

Ajoutons à tout cela, le comportement présent dans le code source du jeu [Figure 6.10], à savoir la colère, celle-ci génère une vitesse plus élevée que toutes les autres chez l'agent autonome, et une couleur noire(définie par moi), en plus de cela, elle change le centre d'intérêt du joueur en terme de démarche, jusque-là ce dernier se déplaçait de manière hasardeuse grâce à la fonction RandomWalk(), une fois le sentiment de colère déclenché, son point d'arrivée se rattache à la position du joueur(comme le montre la figure 6.11 sur la page 32).

```
// Si le tireur atteint l'une des cibles
// - les cibles passent en état de colère ( couleur rouge ) et contrattaque
if (angry && player != null){
    agent.CalculatePath(player.position, path);

    if(path.status != NavMeshPathStatus.PathComplete){
        rend.material.color = normalColor;

        if(anim.GetInteger("State") != 1)
            ContinueWalking();

        RandomWalk();

        return;
    }
```

FIGURE 6.10 – Code source implémentant la réaction d'un agent en état de colère

6.6 Résultat général (positif/négatifs)

6.6.1 Résultats positifs

Les résultats suites à ces tests sont plutôt concluants, et confirment l'hypothèse mise en place par le modèle BDI.

À chaque changement de l'état du monde, rapprochement ou éloignement de l'agent représentant la menace, les paramètres considérées, de joie, de peur, de tristesse ou de colère changent et influent sur la réaction des agents et leur comportement.

FIGURE 6.11 – Visualisation d'agents en état de colère

Le passage entre différentes émotions fonctionne quant à lui aussi très bien, par exemple, si le joueur se rapproche beaucoup des agents, ceux-ci prennent peur et changent leurs démarches afin d'aller plus vite et de fuir, mais si l'agent s'éloigne d'eux, les agents reprennent petit à petit une démarche normale (jusqu'à ce que la distance les séparant soit assez élevé pour qu'ils puissent considérer qu'il n'y a plus de danger), ainsi qu'une couleur correspondant à leur nouvelle émotion.

On peut remarque les prises de décisions sont donc peu "rationnels", et plus "naturels", même si quelques points négatifs restent observables.

6.6.2 Résultats négatifs

L'un des points négatifs peut se résumer dans le passage d'une émotion à une autre qui n'est pas à mon sens pas assez réaliste, en effet dans la réalité, un agent humain peut mettre plusieurs minutes, plusieurs heures voient plusieurs jours ou mois pour se passer d'une émotion à une autre, ou se débarrasser d'une émotion désagréable liée à des situations particulières telles que des traumatismes.

Pouvoir appliquer cela dans un jeu vidéo, implique d'après moi d'ajouter des facteurs temporels à chaque émotion, facteur temporel d'acquisition de l'émotion et facteur temporel de disparition(dismiss) de l'émotion,

ainsi que des facteurs d'intensité, en effet, les études sur les êtres humains montrent que toutes les émotions ne se valent pas en terme d'intensité, de manifestation physique ou psychologique.

Chapitre 7

Conclusion

Pour conclure, nous pouvons constater à partir des recherches faites sur les SMA (systèmes multi-agents) ainsi que leurs domaines d'application suivant différents modèles de programmation telle que le modèle BDI (Belief, Desire, Intention) (Exemple du "SWARMM") que la plupart des agents autonomes actuellement en opération procèdent de manière très rationnelle dans leur prise de décision.

Cela se fait suite à une sélection de multiples possibilités, et élection de celle ayant le plus haut "score" qui est aussi celle-ci ayant prouvé son fonctionnement dans les situations précédentes équivalentes à celle en cours d'exécution.

Ainsi, différents chercheurs se sont penchés sur la question afin de rendre ces agents plus réalistes et crédible aux yeux de l'homme, leurs recherches, d'abord sur l'homme ont prouvé que ce dernier n'effectue un choix rationnel dans une situation donnée au sein d'un environnement donné que très rarement, ils ont en conclue différents modèles dit "Naturels" ou NDM "prise de décisions en milieu naturel" basés sur le comportement des agents humains, notamment ceux qui sont les plus compétents dans le domaine étudié, de manière individuelle ou en groupes, dans un environnement dynamique, rapide et incertain.

Le but était donc ensuite de pouvoir appliquer ces mêmes schémas de pensées et de sélections à des agents autonomes, de nombreux modèles ont en effet vu le jour suite à ces recherches, principalement destinés à l'aide à la décision, ou pour développer de meilleurs modèles cognitifs pour les humains lors des simulations.

Ma contribution dans ce domaine a montré qu'il était possible d'établir certaines règles simples dans un environnement donné constitué d'agents

autonomes, qui permettent de simuler des émotions, basées sur des facteurs physiques, de distance, temporels ou encore de prédation, ces règles peuvent influer sur le comportement de ces agents de manière concrète, c'est-à-dire engendrer des actions suite à des changements dans leur environnement, changements qui influent sur leur état ou émotion du moment et les poussent à repenser leur monde où en tout cas, la vision qu'ils s'en font, et donc de penser un nouveau plan d'action selon l'évolution de leurs émotions.

L'exemple que j'ai pris peut-être très facilement amélioré en intégrant des données plus détaillées sur les émotions comme les facteurs de temps, qui peuvent être issus de la psychologie humaine, ou encore les facteurs d'intensités, qui peuvent être déduits d'expériences issues de situations réelles. ce qui rendrait le passage d'une émotion à une autre plus réaliste et donc plus crédible à l'œil de l'homme.

Annexe A

Code source de personnages non jouables du jeu "Do Not Shoot Aliens"

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;

public class Enemy : MonoBehaviour {

   public NavMeshAgent agent;
   public Animator anim;
   public GameObject spawnEffect;
   public GameObject ragdoll;
   public GameObject bloodEffect;

   public float spawnHeight;
   public float spawnMoveSpeed;
   public float jumpAttackDistance;

    #Variables de la vitesse dépendantes de l'état de l'émotion
    public float walkspeed;
    public float angryspeed;
    public float joyspeed;
    public float sadspeed;
    public float fearspeed;

    #Couleurs dépendantes de l'émotion ( selon la roue des émotions
        de Robert Plutchik)
    public Color normalColor;
```

```
 public Color angryColor; #Rouge
 public Color joyColor; #Jaune
 public Color sadColor; #Bleu
 public Color fearColor; #Vert

 Transform player;
MoveArea area;
GameManager manager;

 #Les émotions primaires
 bool spawned;
 bool angry;
 bool joy;
 bool sad;
 bool fear;

 Vector3 randomTarget;
NavMeshPath path;
Renderer rend;

void Start(){
   PlayerController controller =
       GameObject.FindObjectOfType<PlayerController>();

   if(controller != null)
      player = controller.gameObject.transform;

   area = GameObject.FindObjectOfType<MoveArea>();
   manager = GameObject.FindObjectOfType<GameManager>();

   rend = GetComponentInChildren<Renderer>();
   rend.material.color = normalColor;

   path = new NavMeshPath();

   agent.speed = walkspeed;
   agent.enabled = false;

   Instantiate(spawnEffect, transform.position,
       transform.rotation);
   transform.Translate(Vector3.up * spawnHeight);
}

void Update(){
```

```
if(!spawned){
   transform.Translate(Vector3.up * Time.deltaTime *
       -spawnMoveSpeed);

   if(transform.position.y <= 0){
      transform.position = new Vector3(transform.position.x, 0,
          transform.position.z);
      spawned = true;
      agent.enabled = true;

      anim.SetInteger("State", 1);
      randomTarget = area.RandomPosition();
   }

   return;
}

  # Si le tireur atteint l'une des cibles,
  #les cibles passent en état de colère ( couleur rouge ) et
     contrattaquent
  if (angry && player != null){
   agent.CalculatePath(player.position, path);

   if(path.status != NavMeshPathStatus.PathComplete){
      rend.material.color = normalColor;

      if(anim.GetInteger("State") != 1)
         ContinueWalking();

      RandomWalk();

      return;
   }

      rend.material.color = angryColor;
      agent.destination = player.position;

   if(anim.GetInteger("State") != 2){
      anim.SetInteger("State", 2);
          agent.speed = angryspeed;
          agent.stoppingDistance = jumpAttackDistance;
   }
```

```
 if(Vector3.Distance(transform.position, player.position) <
     agent.stoppingDistance + 0.1f){
    agent.isStopped = true;
    transform.LookAt(player.position);
    anim.SetInteger("State", 3);
    spawned = false;

    StartCoroutine(Attack());
 }



}

# Si la distance entre le tireur et les cibles est assez
#élevée, les cibles passent en état de joie ( couleur jaune
    ) et diminuent leur vitesse
else if (Vector3.Distance(transform.position,
    player.position) > agent.stoppingDistance +20f)
{
    rend.material.color = joyColor;
    agent.speed = joyspeed;
    RandomWalk();

}

# Si le tireur se rapproche des cibles ,
#les cibles passent en état de tristesse(couleur bleue) et
    fuient
else if (Vector3.Distance(transform.position,
    player.position) > agent.stoppingDistance + 14f)
{
    rend.material.color = sadColor;
    agent.speed = sadspeed;
    RandomWalk();
}

# Si le tireur se rapproche des cibles ,
#les cibles passent en état de peur (couleur rouge) et
    fuient (augmentent leur vitesse)
else if (Vector3.Distance(transform.position,
    player.position) > agent.stoppingDistance + 8f)
{
    rend.material.color = fearColor;
```

```
            agent.speed = fearspeed;
            RandomWalk();
        }




        else
        {
            ContinueWalking();
    }
}

void ContinueWalking(){
    anim.SetInteger("State", 1);
    randomTarget = area.RandomPosition();
    agent.speed = walkspeed;
    agent.isStopped = false;
    spawned = true;
}

void RandomWalk(){
    if(Vector3.Distance(transform.position, randomTarget) <
        agent.stoppingDistance + 0.1f){
       randomTarget = area.RandomPosition();
    }
    else{
       agent.destination = randomTarget;
    }
}

public void Hit(){
    Instantiate(bloodEffect, transform.position + Vector3.up *
        1.5f, transform.rotation);
      if (angry){
       Die();
          # Si l'une des cibles est éliminée, les autres passent en
              état de tristesse
      }
    else{
       angry = true;
    }
}
```

```
   void Die(){
      GameObject newRagdoll = Instantiate(ragdoll,
          transform.position, transform.rotation);
      newRagdoll.GetComponentInChildren<Renderer>().material.color =
          angryColor;

        Destroy(gameObject);

    }

   IEnumerator Attack(){
      yield return new WaitForSeconds(0.5f);

      if(player != null && Vector3.Distance(transform.position,
          player.position) > agent.stoppingDistance + 0.1f){
         agent.isStopped = false;
         anim.SetInteger("State", 2);
         spawned = true;
      }
      else{
         manager.GameOver();
         ContinueWalking();
      }
   }
}
```

Bibliographie

Agent Oriented Software JACK. URL : http://agent-software.com.au.

ARBIB, Fellous (2005). *Who Needs Emotions The Brain Meets the Robot*. Oxford University Press. ISBN : 0-19-516619-1,978-0-19-516619-4. URL : http://gen.lib.rus.ec/book/index.php?md5=F2B79880159A912858F365D6B1C1CC1F.

BRATMAN, Michael (1987). *Intention, plans, and practical reason*. T. 10. Harvard University Press Cambridge, MA.

BRATMAN, Michael E, David J ISRAEL et Martha E POLLACK (1988). « Plans and resource-bounded practical reasoning ». In : *Computational intelligence* 4.3, p. 349–355.

D'INVERNO, Mark et al. (1997). « A formal specification of dMARS ». In : *International Workshop on Agent Theories, Architectures, and Languages*. Springer, p. 155–176.

DREYFUS, Hubert L (2014). « Intuitive, deliberative, and calculative models of expert performance ». In : *Naturalistic decision making*. Psychology Press, p. 37–48.

Fostering AI Research : Meet us at AAAI-19, (2019). URL : https://blogs.unity3d.com/2019/01/18/fostering-ai-research-meet-us-at-aaai-19/.

Introducing : Unity Machine Learning Agents Toolkit, (2017). URL : https://blogs.unity3d.com/2017/09/19/introducing-unity-machine-learning-agents/.

KESTEREN, AJ (2001). « A supervised machine-learning approach to artificial emotions ». In : *Department of Computer Science, University of Twente*.

KLEIN, Gary A (2017). *Sources of power : How people make decisions*. MIT press.

LEONHARD, Gerd (2016). *Technology Vs. Humanity : The Coming Clash Between Man and Machine*. Fast Future Publishing.

LIPSHITZ, Raanan (1993). « Converging themes in the study of decision making in realistic settings ». In :

MYERS, David G (2004). « Theories of emotion ». In : *Psychology : Seventh Edition, New York, NY : Worth Publishers* 500.

NORLING, Emma, Liz SONENBERG et Ralph RÖNNQUIST (2000). « Enhancing multi-agent based simulation with human-like decision making

strategies ». In : *International Workshop on Multi-Agent Systems and Agent-Based Simulation*. Springer, p. 214–228.

ORASANU, Judith et Terry CONNOLLY (1993). « The reinvention of decision making ». In :

Orbitofrontal Cortex. https://en.wikipedia.org/wiki/Orbitofrontal_cortex.

Robert Plutchik. URL : https://fr.wikipedia.org/wiki/Robert_Plutchik.

SLOMAN, Aaron, Ron CHRISLEY et Matthias SCHEUTZ (2005). « The architectural basis of affective states and processes ». In : *Who needs emotions* 32.

SMA. URL : https://en.wikipedia.org/wiki/Multi-agent_system.

TAYARI, Imen, Nhan LE THANH et Chokri Ben AMAR (2009). « Modelisation des etats emotionnels par un espace vectoriel multidimensionnel ». In :

ZADEH, Saman Harati, Saeed Bagheri SHOURAKI et Ramin HALAVATI (2006). « Emotional behavior : A resource management approach ». In : *Adaptive Behavior* 14.4, p. 357–380.

ZSAMBOK, Caroline E (2014). « Naturalistic decision making : where are we now ? » In : *Naturalistic decision making*. Psychology Press, p. 23–36.