UNIVERSITÉ PARIS NANTERRE

MÉMOIRE DE MASTER 2 MIAGE

Intégration de comportements humains dans les systèmes autonomes

---

Auteur :
Ana GVIMRADZE

Tuteur :
Lom Messan HILLAH

8 Juillet, 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 mon tuteur ainsi que tout le corps professorial du Master MIAGE, ces deux années ont été riches en expériences.

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

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 agents autonomes auxquels 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 (algorithmes calqués sur le modèle humain et nourris par ce dernier (LEONHARD, 2016)) et de les appliquer à différentes intelligences artificielles dans des jeux vidéo open-source.

1.1 Ordinateurs et émotions

1.1.1 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 ayant acquis le plus de "points" ou "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 (section 2.2) 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 leur application impossible ou difficile.

1.1.2 Objectif concret (framework)

La solution que je propose est un framework développé en 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.

1.2 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).

1.2.1 L'amygdale

L'amygdale (latin, corpus amygdaloideae) représentée sur la figure 1.1, page 4 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. 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 (PILLOU, 2014).

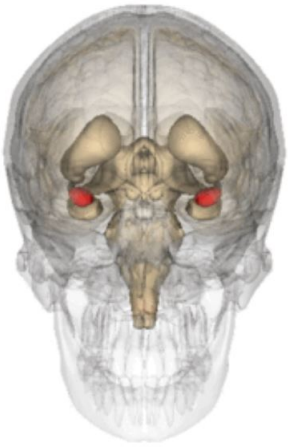

FIGURE 1.1 – Vue 3D de l'amygdale (en rouge).

1.2.2 Le cortex orbitofrontal (OF)

Le cortex orbifrontal qu'on peut observer sur la figure 1.2, page 5 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.

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 (WIKIPÉDIA, 2019a).

1.2.3 Émotions

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

Les émotions se divisent en plusieurs catégories :

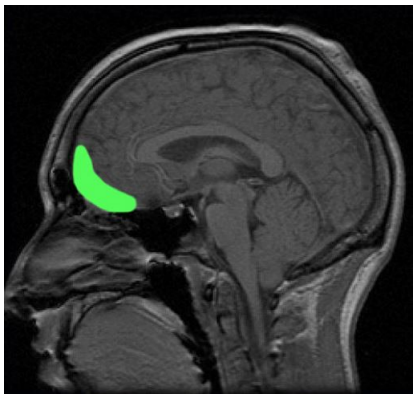

FIGURE 1.2 – Emplacement approximatif de cortex orbitofrontal.

— émotions « cognitives » par opposition aux émotions « non cognitives » ;
— émotions instinctives (des amygdales), par opposition aux émotions cognitives (du cortex orbitofrontal) ;
— émotions primaires et secondaires.

"La théorie psycho-évolutionniste des émotions du professeur et psychologue américain Robert Plutchik (WIKIPÉDIA, 2019c) 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. 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 (LILA, 2012), 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

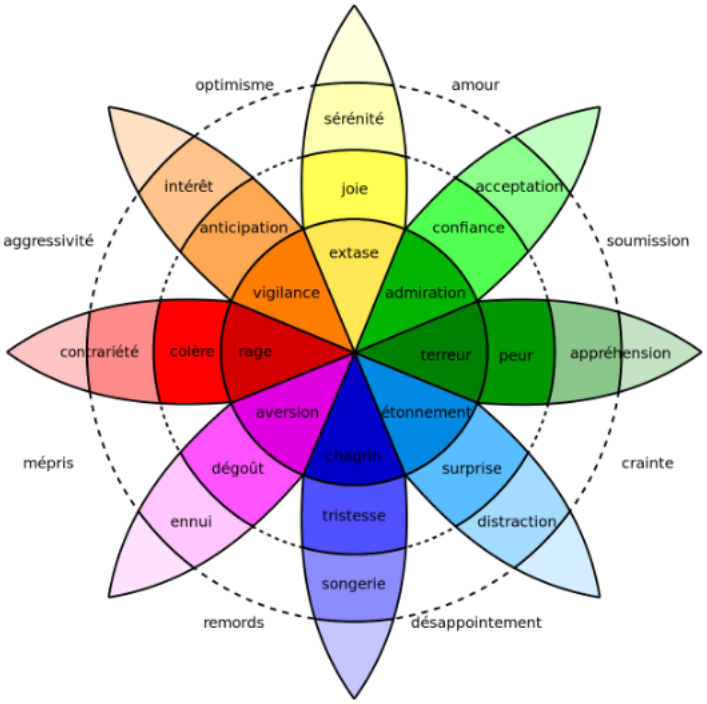

FIGURE 1.3 – Roue des émotions de Robert Plutchik

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 niveaux 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 illustrées par la figure 1.4, page 7 (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) (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

État de l'art

Dans ce chapitre, nous allons voir les principaux concepts impliqués dans l'implémentation d'agents autonomes ; les systèmes multiagents avec quelques exemples d'implémentation ; la recherche en "NDM" ou prise de décisions en milieu naturel et quelques modèles issus de ces recherches. Aussi, l'architecture BDI (belief, desire, intention) servant à l'implémentation d'agents autonomes et qui représentent l'un des modèles les plus proches de la cognition humaine, ainsi que d'autres architectures. Nous verrons aussi comment est abordée l'intelligence artificielle dans les jeux vidéo et de quelle manière elle a évolué depuis l'apparition du jeu vidéo ainsi que deux exemples de jeux célèbres implémentant des IA d'une grande complexité.

2.1 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 interagissants selon certaines relations. Un agent est une entité caractérisée par le fait qu'elle est autonome, ou du moins partiellement. 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 (WIKIPÉDIA, 2019d).

2.1.1 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 et 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, leur environnement, les objets qui les composent ou encore les autres agents ; on les identifie comme étant "les membres de la famille". On intègre ensuite le concept d'organisation, constituée 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) et de capteurs qui permettent aux agents de connaître les changements d'états de leur 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 (WIKIPÉDIA, 2019d).

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

Le programme "SWARMM" (JONES et al., 1999) 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 (D'INVERNO et al., 1997), un environnement de programmation basé sur le modèle BDI (Belief-desire-intention ou Croyance-désir-intention) et implémenté en FORTRAN et en C. Chacun d'entre eux reçoit des données des modèles physiques équivalentes à celles qu'un pilote reçoit de sa vision et de ses instruments et effectue 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 briefings 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 standards et de manière très documentée. Il permet aussi aux combattants de décrire facilement leurs connaissances en se basant sur les différentes étapes du cycle. Pour finir, il assure un échange simple compris entre les deux partis, pilotes et programmeurs. Les programmeurs peuvent ainsi décrire leurs résultats et débattre avec les pilotes en utilisant des termes similaires. Ce programme 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).

2.2 Naturalistic Decision Making (NDM)

> "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 prise de décisions en milieu naturel. De nombreuses études de prise de décisions 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'actions / de rétroactions ;
— le stress dû au temps ;
— des enjeux trop élevés ;
— plusieurs joueurs ;
— objectifs organisationnels et normes.

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

2.2.1 Modèles de NDM

Plusieurs modèles de prise de décisions en milieu naturel 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 être non contradictoires, mais illustrent les différents types de prise de décision qui peuvent être utilisés.

Notez que les chercheurs en NDM s'intéressent généralement aux personnes qui sont expérimentées dans leurs domaines. 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 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 2.1.

Comme le souligne Klein lui-même «Le modèle RPD n'est pas issu 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 (figure 2.1). 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. 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. 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.

Le modèle de la figure 2.1 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.

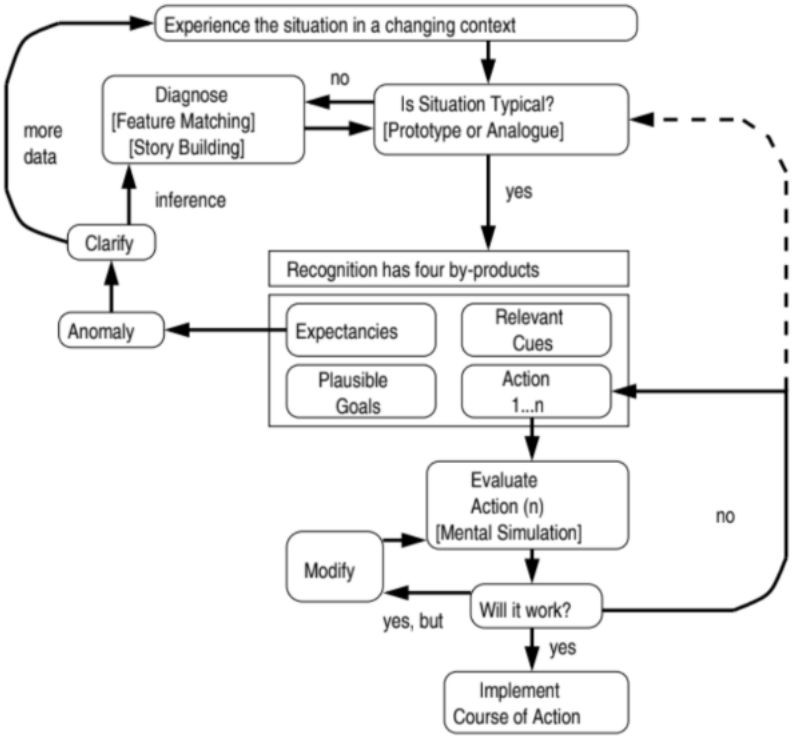

FIGURE 2.1 – La version intégrée du modèle de Klein de décision axée sur la reconnaissance (Figure 7.1 du livre "Sources of Power" by G. Klein 1998 (KLEIN, 2017))

2.3 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 (MATTAR et LANGE, 2019). Dans son article «Programmer un ordinateur pour jouer aux échecs», Shannon écrit :

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

1. le problème est clairement défini à la fois dans les opérations autorisées (les mouvements) et dans le but ultime (le mat) ;
2. il n'est ni si simple jusqu'à être trivial ni trop difficile ;

3. les échecs sont généralement considérés comme nécessitant une «réflexion» ; une solution à 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 considérable pour la création de programmes informatiques capables de jouer à des jeux de manière aussi habiles que des joueurs humains, parfois même en battant les meilleurs. Shannon a inspiré le travail fondateur d'Arthur Samuel sur Checkers entre 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 recherches heuristiques et les méthodes basées sur l'apprentissage (JULIANI, 2017).

Chinook, un programme de contrôleurs mis au point à l'Université d'Alberta en 1989, a commencé à battre les joueurs humains, et dès 1994, les meilleurs joueurs pouvaient au mieux être à égalité avec la machine. Cette tendance s'est poursuivie avec d'autres jeux de plateau à 2 joueurs tels que le backgammon (avec TD-Gammon de Gerald Tesauro, 1992-2002) et le jeu d'échecs (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 Lee Sedol 18 fois 4 à 1, faisant l'objet du documentaire Netflix, AlphaGo (MATTAR et LANGE, 2019).

2.4 BDI (Belief-Desire-Intention)

Les agents du modèle 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.

2.4.1 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.

2.4.2 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.

2.4.3 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. 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 désir) est en train de changer. Un agent stocke ses plans dans une bibliothèque de plans.

2.4.4 Agent BDI

L'agent montré à la figure 2.2, page 17 (NORLING, SONENBERG et RÖNNQUIST, 2000) passe par un cycle continu de :

1. visualiser son 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 2.1.2, 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.

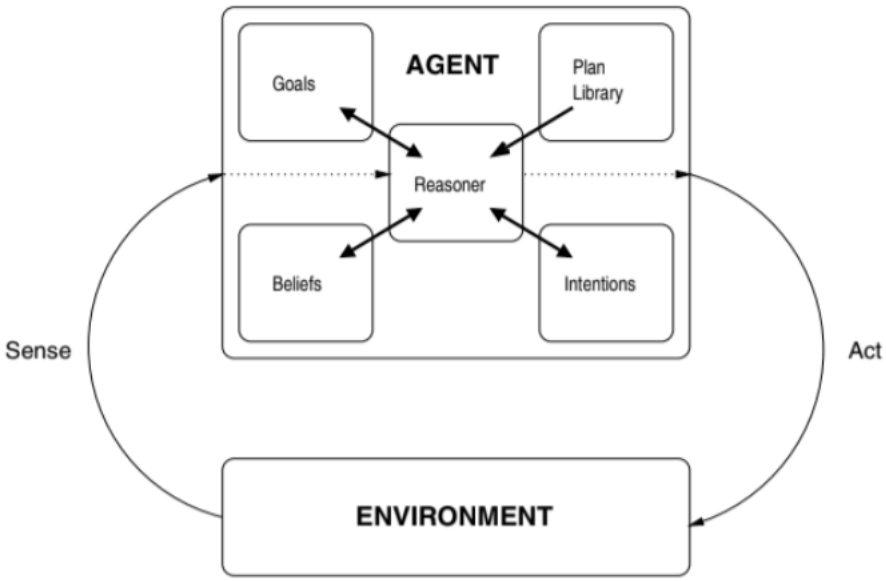

FIGURE 2.2 – Structure d'un agent BDI

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 agents humains 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 (LUCAS, 1997) 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, mais il est également possible d'avoir plusieurs plans applicables dans une situation donnée (lorsque les contextes se chevauchent).

2.5 Autres modèles et architectures

2.5.1 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.

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).

2.5.2 CLARION

CLARION a une structure similaire aux trois niveaux de traitement de l'information (section 2.5.1). C'est une architecture cognitive qui comporte deux niveaux : le niveau implicite (similaire aux niveaux réactifs et de routines) 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 multi-couches avec Q-learning (WATKINS et DAYAN, 1992) 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 méta-cognitif 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.

2.5.3 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 appelés «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 réactif. Il utilise un réseau de neurones pour apprendre comment l'état émotionnel devrait être influencé par la survenue de stimuli.

2.5.4 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 (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 mentales. Ils utilisent uniquement un comportement positif/négatif et un comportement d'approche/d'évitement (uniquement la fonctionnalité du niveau réactif) dans un environnement de type prédateur-proie.

2.6 L'évolution de l'IA dans les jeux vidéo

L'IA n'a pas tant évolué que ça au fil des années, elle fonctionne toujours sur les deux principes de (STATT, 2019) :

— pathfinding : aller d'un point A à un point B, déjà utilisé il y a 20 ans et encore utilisé ;

— state machine : un agent autonome peut être dans les différents états et changer de l'un à l'autre.

L'IA a évolué depuis dans tous les domaines, grâce à un concept clé qui est le deep learning, le fait d'alimenter un agent autonome avec des millions de données issues d'expériences diverses.

Mais les constructeurs de jeux vidéo ne peuvent pas se reposer sur cette technologie, car les agents autonomes basés sur du deep learning ne sont pas prédictibles, et cela pose un grand problème à ces mêmes constructeurs car ils ne peuvent pas prévoir des scénarios précis en utilisant ce type d'IA, c'est pourquoi ils se basent plutôt sur IA ayant des paramètres, qui simulent des comportements aléatoires, mais n'en sont pas vraiment. Simulent, car celles-ci ont des comportements prédéfinis et paramétrés, mais changeants au changement dans leur environnement mais toujours prédictibles, ceux-ci sont toujours basées sur les deux principes fondamentaux cités plus haut.

Une IA qui réagit aux émotions du joueur et construit des plans d'action selon ceux-ci peut sembler improbable, mais la recherche dans ce domaine avance rapidement.

En effet grâce à une technique appelée "génération procédurale" (WIKIPÉDIA, 2019b), popularisée grâce au jeu "No man's sky", cette IA permettait la création d'univers spatiaux totalement aléatoire mais cohérents.

Les recherches à partir de cette technique visent à réaliser des jeux entiers en "génération procédurale", l'univers, les personnages, leurs attitudes vis-à-vis du joueur etc.

Les développeurs de jeux vidéo pourront donc créer, non seulement des univers procéduraux et aléatoires, mais aussi construire une expérience de jeu propre aux goûts de chaque joueur, qui s'améliorera au fil du temps que le joueur passe sur le jeu, ce qui lui permettra de s'alimenter de ses goûts et préférences et de les implémenter en temps réel (STATT, 2019).

Les jeux cités dans les sections 2.6.1 et **??** sont parmi les meilleurs dans l'implémentation d'agents autonomes (personnages non jouables) dans l'industrie du jeu vidéo. C'est aussi, en partie, ce qui fait leur grand succès auprès des joueurs, le fait que tout rappelle le monde réel.

2.6.1 Red Dead Redemption 2

Dans ce jeu doté d'immenses paysages sorti tout droit d'un western, le joueur peut parler à n'importe quel personnage en appuyant sur le déclencheur gauche et en sélectionnant une interaction positive ou négative. Celles-ci en pour conséquence des actions différentes à chaque fois, en fonction du contexte : les vêtements du joueur, les taches de sang, la boue, l'emplacement, ce que fait l'agent autonome au moment de la prise de contact, la côte d'honneur, la quantité d'alcool que le joueur a consommé, et plus encore (MCKEAND, 2018).

Comme on peut le voir ici, les actions des agents ne sont pas complètement aléatoires, elles sont programmées, mais ce qui fait leur force et donne l'illusion du "monde réel ", c'est le nombre épatant de paramètres entrants en compte pour déterminer quelle action va entreprendre l'agent, un petit détail qui change dans leur environnement peut engendrer une modification dans le déroulement d'une situation (un cheval qui passe à une grande vitesse, un autre agent qui dit un gros mot etc.).

Les émotions que possède chaque personnage non jouable ne sont pas nourries par une grande quantité de données comme on peut le trouver dans d'autres secteurs utilisant l'intelligence artificielle, mais par une quantité de conditions qui déterminent les différents états émotionnels, ce qui rend les agents prédictibles et permet aux constructeurs de construire une histoire cohérente, avec une dose de mystère maîtrisée.

En plus de l'implémentation d'une IA complexe, une énorme quantité de gestes et d'animations subtiles l'accompagnent, ce qui élève le jeu entier à un niveau beaucoup plus réaliste.

Par exemple le fait que tous les dialogues se passent en temps réel dans le jeu plutôt que dans une liste d'options de dialogue sur un écran statique. C'est une nouvelle approche de l'interaction dans le jeu - déclenchée de la même manière que vous tirez avec une arme à feu sur un agent autonome - et vous donne l'illusion que tout est possible. Plutôt que d'interagir avec ce monde en tuant et en mutilant, Red Dead Redemption 2 donne des mots aussi puissants que des balles.

Ou encore, "si un témoin vous surprend en train de faire quelque chose, Arthur (le personnage joué par l'utilisateur) dit : " Ah, merde ... " car vous devez les pourchasser", ajoute David Hynd, programmeur en charge de l'IA. "Ou bien, si vous pénétrez dans un salon, l'IA peut s'arrêter dans sa

tâche et la musique peut cesser de jouer - l'IA vous regardera un instant avant de revenir sur ses affaires. Ou pas, s'ils vous voient comme une menace. Et c'est incroyable de voir comment Arthur peut chanter et fredonner, ou comment on peut transformer le cours des choses en interagissant avec les agents autonomes quand il se saoule".

Je pense que ce sont les petits détails qui font que tout paraît si réaliste.

2.6.2 The Sims 4

Le 2 septembre 2014, le studio Maxis d'Electronic Arts lancera The Sims 4. Comme toute suite, le jeu ressemblera à son vieil ami : on peut toujours créer nos propres "sims", construire notre propre maison, déterminer notre propre vie et, espérons-le, ne pas tuer ni négliger notre propre avatar. Mais la technologie sous-jacente - les os et les nerfs qui maintiennent l'univers ensemble - fait actuellement l'objet d'une mise à niveau majeure pour la quatrième génération (ORF, 2014).

L'une des avancées à consister à peaufiner le système de routage, ou la façon dont les sims naviguent. Après avoir étudié le comportement social humain, les développeurs ont rendu les sims moins gênants et plus crédibles. "Les sims peuvent maintenant franchir les portes sans rester coincés, mais il est extrêmement difficile d'y arriver" dit Pearson Ingebretson - Ingénieur logiciel pour the Sims 4.

Au cœur de chaque jeu des sims se trouve son intelligence artificielle, son réseau infini de modèles (figure 2.3) et ses résultats possibles qui font tourner le monde en coulisses. Dans les jeux, l'intelligence artificielle fait généralement référence à la manière dont l'environnement, et en particulier les personnages non joueurs, interagissent avec le monde et le joueur. Cependant, la franchise Les Sims se distingue.

"Dans de nombreux autres jeux, l'IA est étrangère au joueur quant à la manière dont elle se comporte. L'IA commande aux personnages de faire des choses fondamentalement différentes de celles des joueurs ... Ils opèrent dans un monde différent avec des possibilités différentes", explique Ingebretson. "Dans Les Sims, si vous restez assis à regarder votre ordinateur pendant un moment, l'IA prendra le contrôle et contrôlera également les actions de vos sims. Cela signifie que notre IA doit prendre des décisions plus crédibles afin que le cours de l'action reste cohérent "(ORF, 2014).

Écouter Ingebretson qui décrit les tenants et les aboutissants de l'intelligence artificielle dans The Sims 4 ressemble à un cours de psychologie humaine. À la base, l'IA travaille avec l'interaction de deux mécanismes : les commodités et les courbes d'utilité. Les commodités représentent l'état interne d'un Sim, tandis qu'une courbe d'utilité dicte le désir d'un avatar à améliorer ses commodités. Chaque interaction ouvre une série d'améliorations possibles. "Par exemple, si un Sim boit une tasse de café, son énergie va monter mais sa vessie va baisser", dit Ingebretson. Comme dans le monde réel, tout est un compromis. Ainsi, lorsqu'il prend chaque décision, un Sim considère toutes les actions possibles, analyse leurs résultats, référence la courbe d'utilité et sélectionne la meilleure. Cependant, ces actions sont quelque peu aléatoires de par leur conception, de sorte que l'IA puisser donner l'illusion qu'elle est imprévisible.

Cet échange en une fraction de seconde entre la prise de contrôle de l'IA et du joueur est ce que Ingebretson appelle «autonomie», ou lorsqu'un sim est en pilote automatique. Dans les jeux précédents des Sims, l'autonomie était un système de départ et d'arrêt. Une fois qu'un sim a terminé avec une action, il s'exécute de manière autonome. Dans Les Sims 4, les développeurs ont amélioré l'efficacité de l'IA en créant une hiérarchie d'autonomie. "Au lieu de tout considérer dans le monde entier chaque fois qu'un Sim décide quoi faire, nous évaluons d'abord tous les produits de base et déterminons quels types d'actions sont les plus importantes pour le Sim", explique Ingebretson. "Cela nous permet d'éliminer de nombreuses possibilités." Cela signifie que les Sims deviennent plus rapides et plus efficaces pour prendre des décisions et peuvent effectuer plusieurs tâches plutôt que de suivre un script strict «d'abord ceci, ensuite cela» (ORF, 2014).

Toutes ces améliorations et ces avancées rendent un jeu plus naturel, mais quand est-ce qu'un système d'intelligence artificielle devient-il trop efficace ? "C'est un sujet intéressant que nous traitons dans chacun de ces projets, car nous pouvons continuer à rendre les Sims suffisamment intelligents pour mener leur propre vie", a déclaré Pearson. "Qu'est-ce qui est trop intelligent ?"

Laissée à ses propres appareils parfaitement optimisés, l'IA pourrait facilement prendre le contrôle et jouer au jeu pour vous, laissant ainsi aux Sims effectuer un nombre anormal d'actions en même temps ou annuler

l'interaction du joueur. L'équipe a introduit des seuils de retard et d'attention pour limiter artificiellement le système. "Il est impossible pour un cerveau humain de gérer autant de flux d'entrées", explique Ingebretson. "Pour les Sims, nous construisons un modèle de la vie humaine. Nous devons modéliser les fautes des gens ainsi que leur efficacité" (ORF, 2014).

Nous pouvons apercevoir que l'approche dans Les Sims 4 afin de rendre l'IA plus crédible est différente du cas précédent, même si les deux approches ont des similitudes dans la complexité de leur modèle, qui consiste à alimenter les personnages d'un grand nombre de conditions détaillées et en cascade (un geste, une quantité d'alcool consommée, une intonation...), avec des scénarios différents pour chaque état, c'est tout ces détails qui donnent l'illusion de la réalité.

En plus de cela, Les Sims 4 propose un système tout à fait novateur, qui consiste en la prise de contrôle de l'IA du personnage joué par l'utilisateur lors des moments où celui-ci est au repos (quand le joueur laisse tourner le jeu sans y toucher). Mais celle-ci ne le fait pas n'importe comment, elle le fait tout en restant cohérente avec l'état précédent de l'agent, par exemple si le personnage est en train de boire de l'eau et que l'IA intervient et en prend le contrôle, elle le pousse à laver son verre, avant de l'essuyer de le ranger dans l'évier.

En plus de cela, les fautes des gens ainsi que leurs moments d'inattention et de faiblesse ont aussi été modélisés, non pas de manière directe, mais en introduisant des seuils de retards dans la succession des actions, afin que celles-ci aient l'air naturelles, et qu'elles ne ressemblent pas à un flot ininterrompu d'actions, comportement qui n'est pas compréhensible par l'homme.

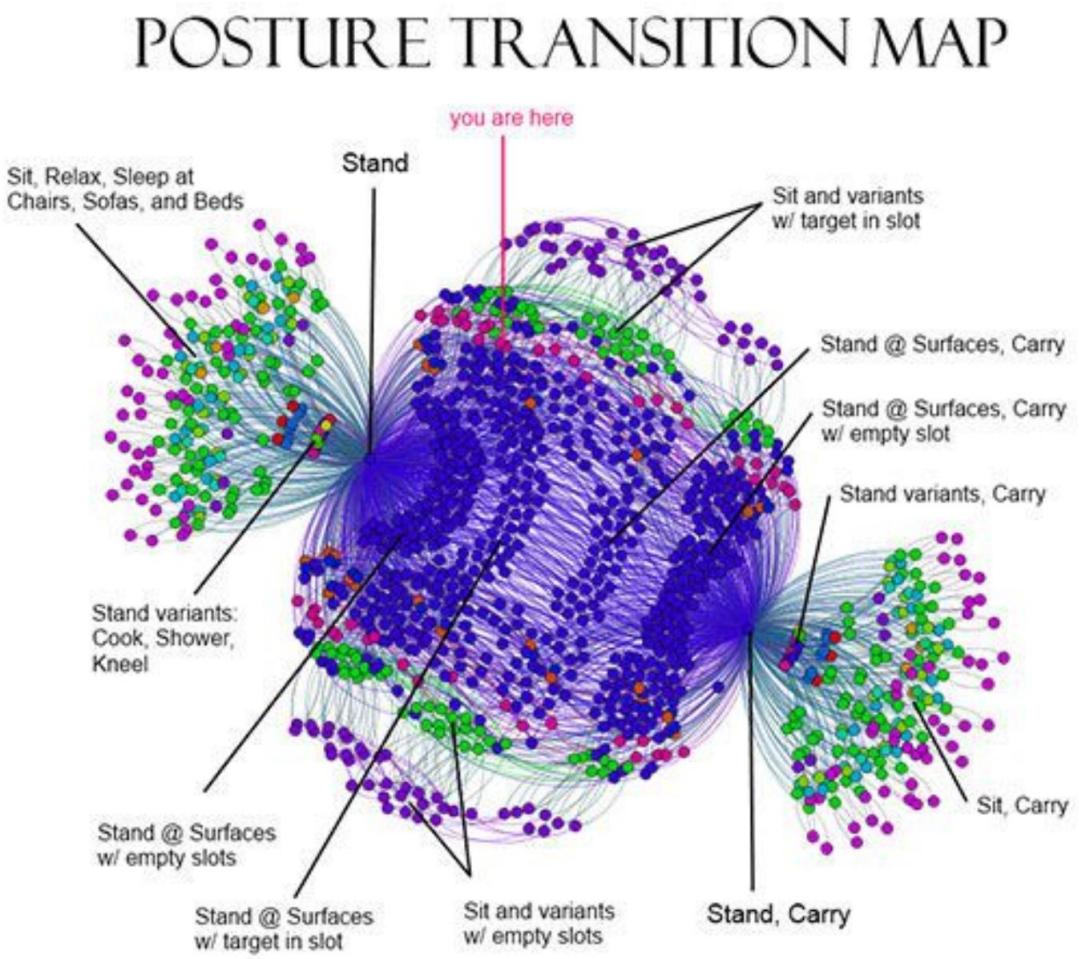

FIGURE 2.3 – Cette image visualise le potentiel des conséquences d'un changement de posture d'un Sim dans Les Sims 4

Chapitre 3

Contribution

3.1 Unity

Unity est le logiciel que j'utilise pour développer mon framework, c'est est un moteur de jeu multi-plateforme (smartphone, ordinateur, consoles de jeux vidéo et Web) développé par Unity Technologies (JULIANI, 2017). 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 permette 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 ce dernier selon des paramètres précis, objectifs, actions et réactions.

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

3.2 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 (figure 3.1).

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.

La victoire est basée sur le fait d'accumuler le plus de points possible en allant de check-point (point de contrôle) en check-point.

FIGURE 3.1 – Icon du jeu "Do Not Shoot Aliens"

3.3 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, ainsi que l'environnement et les objets avec lesquels les agents interagissent, est géré par un code source qui lui est propre, ce dernier est écrit en C# et associé à l'agent concerné via l'interface graphique de Unity, l'avantage de ce langage compilé est le gain de performances.

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 aux concepts de classe, variables de classe et méthodes de classe présentes dans les langages orientés objet.

3.4 Description des agents autonomes présents dans le jeu

Le scénario typique pour la création et 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 (MATTAR et LANGE, 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 circuler sur l'esplanade, en effet, tant qu'ils ne sont pas touchés, ils conservent leur démarche lente et continuent de circuler de manière hasardeuse (grâce à une fonction Random() qui permet de générer les coordonnées de 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 (section 2.4.4) :

— leur "Belief" (ou croyance) initiale est de croire que leur environnement est inoffensif, c'est leur vision du monde ;
— ce qui engendre leurs désires ou encore leurs objectifs, 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 agents différents paramètres qui leur permettront de réagir à leur nouvel environnement.

3.5 Application de paramètres de prise de décisions en milieu naturel (NDM) aux agents BDI

Afin d'appliquer ces paramètres, je procède de la manière suivante (figure 3.2) :

— je déclare des paramètres qui permettent d'évaluer l'état émotionnel des agents lors des différentes phases de changement de l'environnement ;
— ces variables sont : la peur, la joie, la tristesse et la colère qui sont des booléens ;
— ces variables possède à son tour des paramètres qui lui sont propres : la vitesse de la démarche, ainsi qu'une couleur qui est attribuée aux agents lorsqu'il ressentent l'une de ces émotions ;
— 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 3.3, page 31).

```
// 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 3.2 – Déclaration des variables

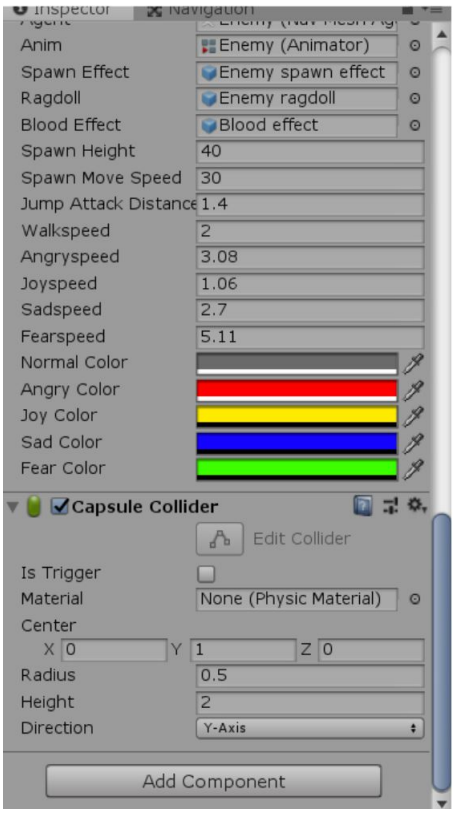

FIGURE 3.3 – Fenêtre Unity, Valeurs des paramètres

3.5.1 Première expérimentation

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

La distance est exprimée ici par la différence entre la position de l'agent en mode "stop" ou à l'arrêt et le joueur lui-même, à laquelle on ajoute un delta, un 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 grande selon le delta) du joueur.

Le résultat souhaité est de pouvoir visualiser un changement de couleur aux agents suivant l'émotion qu'ils ressentent, dans la figure 3.4, c'est la peur, la couleur de l'agent prend donc la couleur associée à la peur (le

```
// 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 3.4 – Code source implémentant une courte distance entre le joueur et l'agent

vert, figure 3.5), 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 élevé lui permettant d'échapper au joueur au fur et à mesure que celui-ci se rapproche. La vitesse liée à la peur est plus élevée que celle liée à la démarche dite "normale", ceci est exprimé sur la figure 3.3.

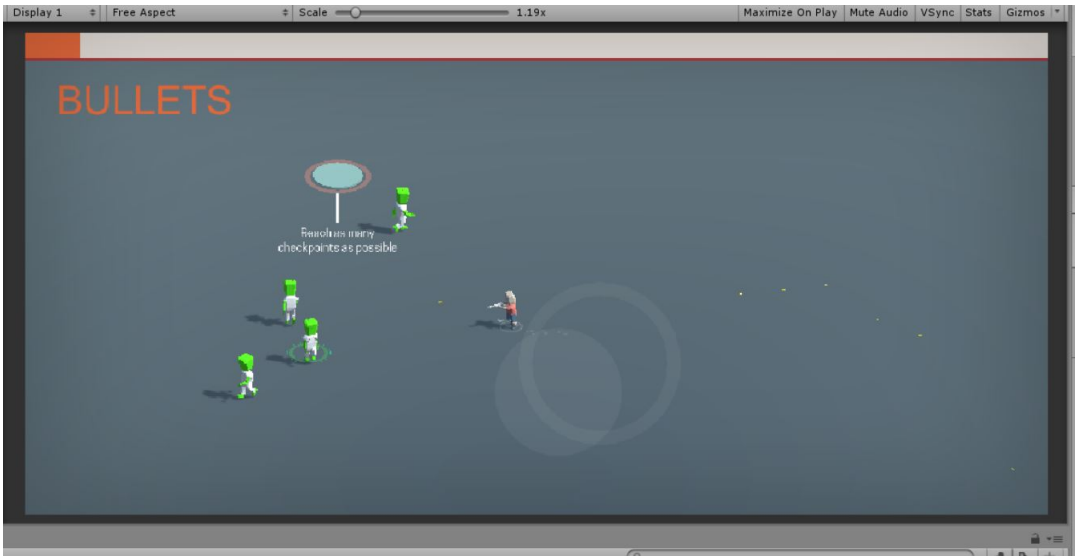

FIGURE 3.5 – Visualisation d'agents en état de peur

3.5.2 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 que l'on considérera comme élevée par rapport à la taille du terrain (figure 3.6).

```
// 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 3.6 – Code source implémentant une longue distance entre le joueur et l'agent

Le résultat souhaité est de pouvoir visualiser un changement de couleur sur les agents à l'instant où cette distance est atteinte.

La couleur de ces derniers devient alors jaune, et leur vitesse passe à un rythme beaucoup moins élevée (figure 3.7).

FIGURE 3.7 – Visualisation d'agents en état de joie

3.5.3 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 3.8).

```
// 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 3.8 – Code source implémentant une distance médiane entre le joueur et l'agent

Le résultat souhaité est de pouvoir visualiser un changement de couleur sur les 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 3.9).

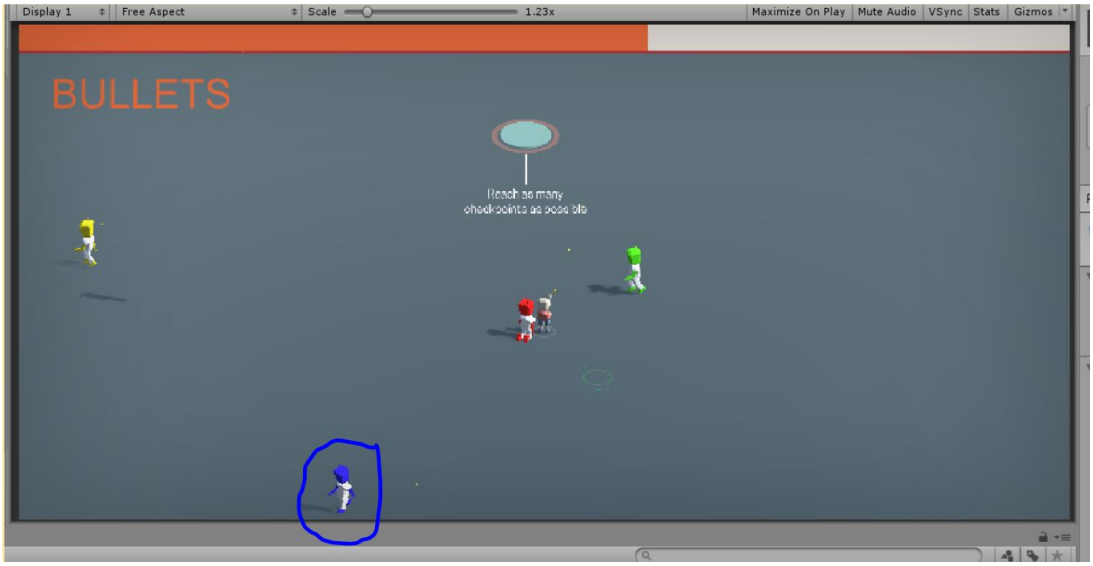

FIGURE 3.9 – Visualisation d'agents en état de tristesse

Ajoutons à cela, le comportement présent dans le code source du jeu (figure 3.10), à savoir que la colère génère une vitesse plus élevée que toutes les autres émotions chez l'agent autonome et une couleur rouge. En plus de cela, elle change le centre d'intérêt de l'agent. En effet, 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 instantanément à la position du joueur (comme le montre la figure 3.11 sur la page 36).

```
// 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 3.10 – Code source implémentant la réaction d'un agent en état de colère

3.6 Résultats (positifs/négatifs)

3.6.1 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 autonome du joueur représentant la menace, les paramètres considérés, de joie, de peur, de tristesse ou de colère changent et influent sur la réaction des agents et leurs comportements.

FIGURE 3.11 – Visualisation d'agents en état de colère

Le passage entre les 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 le joueur s'éloigne d'eux, les agents reprennent petit à petit une démarche normale, en passant par différents états émotionnels, qui sont représentés visuellement par les différentes couleurs, jusqu'à ce que la distance les séparant soit assez élevée pour qu'ils puissent considérer qu'il n'y a plus de danger.

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

3.6.2 Résultats négatifs

L'un des points négatifs est le passage d'une émotion à une autre qui n'est, 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 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 qu'un facteur d'intensité, en effet, les études sur les êtres humains montrent que toutes les émotions ne se valent pas en terme d'intensité lors de leurs manifestations physique ou psychologique.

3.7 Avis utilisateurs

Dans cette section je présenterai les avis que j'ai recueilli auprès des personnes lesquelles ont pu tester le jeu "Do Not Shoot Aliens" sans et avec mes modifications.

Souda : "les IA semblent être en colère lorsque le tireur s'approche d'eux. La colère de l'IA prend le dessus sur la peur. En revanche, on constate que lorsque le tireur est à distance moyenne de l'IA, celui-ci a peur même si le danger se trouve à distance.

Une personne qui passe à côté d'une personne morte ne présente pas d'émotion de tristesse, elle a peur de bien arrêter = sentiment de paralysie.

Les sentiments changent vite, tristesse puis autre émotions. Pas le temps de s'apitoyer sur le triste sort de quelqu'un.

Une personne en colère, sentant sa fin approcher sûrement, va se mettre à poursuivre l'assaillant. L'action de dernière chance pour survivre dans le cadre du jeu.

En réalité on pourrait penser que plus la personne est proche de l'assaillant plus il est paralysé de peur, dans le jeu c'est le contraire.

Dans l'IA on peut faire un parallèle avec les voitures autonomes, qui vont juger dans le futur à notre place qui doit mourrir et qui doit rester en vie en cas d'accident mortel imminant et sûr. Chaque humain réagit différemment et juge de la situation différemment. L'IA réagit différent dans ce jeu aussi. Certains vont essayer de se transformer en héros et d'autres ne luttent pas et se laissent tirer dessus."

Hajar : "comme a dit Souda, chacun réagit différemment devant un danger. Mais, j'ai aussi l'impression que ceux qui sont en colère meurent

les premiers. Cela veut dire qu'ils représentent un danger pour le tireur du coup, il va s'en prendre à eux en premier.

Le tireur se sent menacé par les rouges donc il les tue. Le tireur va d'abord choisir d'éliminer les IA qui représentent un danger pour lui."

Chapitre 4

Conclusion

Pour conclure, nous pouvons constater à partir des recherches faites sur les SMA (systèmes multi-agents) ainsi que leurs domaines d'applications, et les différents modèles de programmation avec lesquels celles-ci sont conçues, telle que le modèle BDI (section 2.4.4) et l'exemple du "SWARMM" (2.1.2) que la plupart des agents autonomes actuellement en opération procèdent de manière très rationnelle dans leur prise de décisions.

Cela se fait suite à une sélection de multiples possibilités, et élection de celle ayant le plus haut "score" qui est aussi celle 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 sur lequel le modèle est appliqué (médical, militaire, aéronautique, etc.).

Le but était ensuite de pouvoir appliquer ces mêmes schémas de pensées et de sélections à des agents autonomes, de nombreux modèles ont vu le jour suite à ces recherches, principalement destinés à l'aide à la décision, ou pour développer de meilleurs modèles cognitifs pour ces agents 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 des facteurs de temps, qui peuvent être issus de la psychologie humaine lors de recherches sur les émotions sur ce dernier, ou encore des facteurs d'intensités, qui peuvent être déduits 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.

Selon les retours de Souda et d'Hajar que j'ai pu avoir, je peux conclure que ma contribution pour le jeu vidéo "Do not shoot Aliens" a eu un résultat. Même si c'est loin d'être parfait, nous avons déjà pu obtenir des idées d'utilisateurs pour les axes d'amélioration.

Le plus important, à mon avis, est qu'on a pu voir très clairement qu'en ajoutant des critères plus en plus précises, il y a la possibilité de simuler le comportement des êtres humains avec les IA.

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

ARBIB, Fellous (2005). *Who Needs Emotions The Brain Meets the Robot*. Oxford University Press. 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.

JONES, Randolph M et al. (1999). « Automated intelligent pilots for combat flight simulation ». In : *AI magazine* 20.1, p. 27–41.

JULIANI, Arthur (2017). *Introducing : Unity Machine Learning Agents Toolkit*, 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.

LILA, Florence (2012). *Les émotions fondamentales d'après Paul EKMAN*. URL : http://psychologie-des-emotions.eklablog.com/les-emotions-fondamentales-d-apres-paul-ekman-a46035086.

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

LUCAS, Andrew (1997). *Agent Oriented Software JACK*. URL : http://agent-software.com.au/jack.html.

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

MCKEAND, Kirk (2018). *Red Dead Redemption 2 – how advanced AI and physics create the most believable open world yet*. URL : https://www.vg247.com/2018/12/12/red-dead-redemption-2-physics-ai-euphoria-phil-hooker-interview/.

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 : *Ablex Publishing*.

ORF, Darren (2014). *The AI That Powers The Sims 4 Is Almost Too Smart*. URL : https://www.popularmechanics.com/culture/gaming/a10698/inside-the-mind-of-the-sims-4-16906802/.

PILLOU, Jean-François (2014). *Amygdale*. URL : https://sante-medecine.journaldesfemmes.fr/faq/21535-amygdale-cerveau-definition.

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

STATT, Nick (2019). *How AI will completely change video games*. Youtube. URL : https://www.youtube.com/watch?time_continue=95&v=NPuYtHZud0o.

TAYARI, Imen, Nhan LE THANH et Chokri Ben AMAR (2009). « Modelisation des etats emotionnels par un espace vectoriel multidimensionnel ». In : *Technical report, Laboratoire Informatique, Signaux et Systemes de Sophia*.

WATKINS, Christopher JCH et Peter DAYAN (1992). « Q-learning ». In : *Machine learning* 8.3-4, p. 279–292.

WIKIPÉDIA (2019a). *Cortex orbitofrontal — Wikipédia, l'encyclopédie libre*. [En ligne ; Page disponible le 19-mars-2019]. URL : http://fr.wikipedia.org/w/index.php?title=Cortex_orbitofrontal&oldid=157678148.

— (2019b). *Génération procédurale — Wikipédia, l'encyclopédie libre*. [En ligne ; Page disponible le 2-mai-2019]. URL : http://fr.wikipedia.org/w/index.php?title=G%C3%A9n%C3%A9ration_proc%C3%A9durale&oldid=158928549.

— (2019c). *Robert Plutchik — Wikipédia, l'encyclopédie libre*. [En ligne ; Page disponible le 21-mars-2019]. URL : http://fr.wikipedia.org/w/index.php?title=Robert_Plutchik&oldid=157735068.
— (2019d). *Système multi-agents — Wikipédia, l'encyclopédie libre*. [En ligne ; Page disponible le 20-mai-2019]. URL : http://fr.wikipedia.org/w/index.php?title=Syst%C3%A8me_multi-agents&oldid=159422109.
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.