Mémoire de fin d’études

L’intelligence artificielle appliquée aux personnages non-joueurs dans les jeux vidéo

Comment rendre l’environnement d’un jeu intelligent ? Dans quel but ?

Sébastien Gaulier - 5A IJV

Guillaume Ambrois - 5A IJV

Maître de mémoire : M. Alain Lioret



**5A 2015**

# Sommaire

[1. Sommaire 1](#_Toc443952264)

[2. Avant-propos 2](#_Toc443952265)

[2.1. Remerciements 2](#_Toc443952266)

[2.2. Résumé 3](#_Toc443952267)

[2.3. Abstract 4](#_Toc443952268)

[2.4. Mots clés 5](#_Toc443952269)

[3. Introduction 6](#_Toc443952270)

[4. Bref historique de l’IA appliquée aux jeux vidéo 7](#_Toc443952271)

[4.1. La naissance de l’IA 8](#_Toc443952272)

[4.2. Les premières IA pour le jeu vidéo 9](#_Toc443952273)

[4.3. Une IA évolutive 10](#_Toc443952274)

[5. Les applications possibles prometteuses 12](#_Toc443952275)

[5.1. Le Machine Learning, l’IA qui apprend en même temps que le joueur 12](#_Toc443952276)

[5.1.1. Les réseaux de neurones 12](#_Toc443952277)

[5.1.2. Les arbres de décision avec l’apprentissage par renforcement 14](#_Toc443952278)

[5.2. Dynamic Game Difficulty Balancing, l’IA qui s’adapte au niveau du joueur 16](#_Toc443952279)

[5.3. Des jeux par les IA, pour les IA 18](#_Toc443952280)

[5.3.1. General Game Playing, l’IA qui apprend à jouer 18](#_Toc443952281)

[5.3.2. Angelina, l’IA qui développe des jeux 19](#_Toc443952282)

[6. De l’importance de l’IA pour l’immersion du joueur 21](#_Toc443952283)

[6.1. Suspension consentie de l'incrédulité, solipsisme et vallée dérangeante 22](#_Toc443952284)

[6.1.1. Solipsisme 23](#_Toc443952285)

[6.1.2. Vallée Dérangeante 25](#_Toc443952286)

[6.2. Entre la sérendipité et l’apophénie 27](#_Toc443952287)

[6.2.1. Fable, un monde plus vivant et cohérent qu’il n’y paraît 28](#_Toc443952288)

[6.2.2. Evie / Cleverbot 31](#_Toc443952289)

[7. Conclusion 32](#_Toc443952290)

[8. Références 33](#_Toc443952291)

# Avant-propos

## Remerciements

Nous tenons à remercier :

M. Alain LIORET, notre maître de mémoire, pour son soutien, sa passion, et pour nous avoir guidé dans notre travail. Il nous a donné de très bonnes pistes de recherches, ainsi que de bons conseils dans nos choix.

L’équipe pédagogique de l’ESGI, pour la qualité des cours dispensés à l’ESGI, qui nous ont donné la motivation nécessaire pour rédiger ce mémoire.

## Résumé

Les jeux vidéo sont de plus en plus beaux, que ce soit en approchant le photoréalisme, ou en ayant une patte graphique unique. Ils sont également de plus en plus profonds, avec une histoire, une narration et un univers toujours plus poussés. Le gameplay également n’a de cesse de s’améliorer, que ce soit à travers de nouvelles idées innovantes de game design, ou via de nouveaux périphériques pour renforcer l’interaction ainsi que l’immersion du joueur.

Mais le domaine dans lequel il reste le plus de progrès à faire pour proposer des expériences toujours plus intéressantes et immersives est l’intelligence artificielle (IA). Bien que celle-ci soit de plus en plus présente (et attendue) dans les jeux, et de plus en plus évoluée grâce à des algorithmes toujours plus performants, cela reste en général une IA « fixe », avec des comportements prévus à l’avance, en réponse à des évènements prévus à l’avance.

En effet, bien souvent les personnages non joueurs (PNJ) sont des entités statiques du jeu ne possédant qu’un panel fixe d’actions ou de répliques de dialogues, qu’ils soient alliés (marchand), ou ennemis (monstre). Ces personnages sont ainsi relégués au rang d’élément du décor, alors qu’ils pourraient occuper une place bien plus importante dans le jeu, apportant à la fois en profondeur de gameplay et de narration.

Dans ce mémoire, nous étudierons les différentes IA qui existent actuellement, ainsi que ce qu’elles apportent en termes d'expérience.

## Abstract

Video games are more and more beautiful, whether it be by approaching photorealism, or by using a unique graphic style. They’re also more and more deep, with a story, a narration and an universe always more advanced. The gameplay keeps getting better, either with innovative game design ideas, or with new devices to enhance the interaction and immersion of the player.

But there is a field in which much progresses are yet to be done in order to offer ever more interesting and immersive experiences, and this field is artificial intelligence (AI). Although it’s more and more present (and expected) in video games, and more and more efficient through algorithms ever more powerful, it remains, in general a scripted AI, with programmed behaviors which respond to specific events.

Indeed, often the non-player characters (NPCs) are static entities of the game with only a fixed panel of actions or dialogues replicas, whether allies (merchant) or enemies (monster). These characters are relegated to the role of decorative elements, when they could take a more important place in the game, providing both depth of gameplay and storytelling.

In this thesis, we will study the state of the art of artificial intelligence applied on NPCs, and what they offer in term of experience.

## Mots clés

|  |  |
| --- | --- |
| Mots clés | Key words |
| Intelligence Artificielle  Jeu Vidéo  Personnage Non Joueur (PNJ)  Apprentissage Automatique  Réseaux de neurones  Arbres de décision | Artificial Intelligence  Video Game  Non Playable Character (NPC)  Machine Learning  Neural Networks  Decision Trees |

# Introduction

Depuis leur invention, les jeux vidéo n’ont eu de cesse de s’améliorer, que ce soit du point de vue du gameplay, du scénario ou du matériel. Mais un aspect qui est reste encore aujourd’hui en retrait (en plus du fait que lorsqu’elle est bien réalisée, elle est discrète), c’est l’intelligence artificielle.

Un premier problème se situe au niveau de la crédibilité du comportement, empêchant le joueur de s'immerger totalement. C’est une problématique impactant principalement les jeux solos. En effet, le cœur des jeux vidéo se jouant seul est son scénario. Il est donc nécessaire que l’histoire ainsi que ses personnages paraissent le plus crédible possible aux yeux du joueur.

Un deuxième problème se situe au niveau du degré de compétences de l’IA. En effet, celui-ci est souvent fixe, défini à l’avance, et impersonnel. Il existe plusieurs solutions pour pallier à ce problème, et si quelques jeux utilisent déjà ce genre d’IA un peu plus évoluée (notamment Left 4 Dead et son AI Director, dont nous parlerons plus en détail plus tard), trop peu sont les jeux à les utiliser.

Le constat est donc simple, pour apporter une meilleure expérience au joueur, il faut améliorer le comportement des agents en le rendant plus réaliste d’une part, et plus personnalisable, adaptable, et évolutif d‘autre part.

Il existe beaucoup de types d’IA applicables pour les jeux vidéo, et beaucoup de façon de les réaliser. Nous présenterons dans ce mémoire les principales méthodes, notamment certaines basées autour du Machine Learning, cependant, pour celles-ci, nous restreindrons le champ d’application aux PNJ.

# Bref historique de l’IA appliquée aux jeux vidéo

Dans les jeux vidéo, l’intelligence artificielle est utilisée pour donner vie aux PNJ. Les méthodes employées pour implémenter les algorithmes de ces PNJ s’appuient généralement sur les théories existantes du domaine plus général de l’intelligence artificielle. Cependant, contrairement à une IA classique comme celle d’un robot, l’objectif n’est pas nécessairement de reproduire le comportement d’un humain. Il est même fréquent de voir des PNJ tricher.

Dans la plupart des jeux, les PNJ ont par exemple connaissance de l’ensemble de la carte et de ce qui s’y déroule. Dans les jeux de type FPS (« First Person Shooter »), les IA sont souvent dotées d’une visée parfaite. Par conséquent, il est très souvent nécessaire de brider ces capacités hors normes pour donner au joueur une sensation d’équité.

L’histoire de l’IA dans ce domaine remonte sans doute au début des années 1940s avec le jeu de stratégie pure **Nim** (publié en 1942). L’IA était alors capable de gagner, même contre des joueurs chevronnés !



Figure : Le jeu de Nim

## La naissance de l’IA

En 1951, les premières IA pour les jeux de Dames et d’Echecs sont créées à partir de la machine Ferranti Mark 1[[1]](#footnote-2). Ces IA ont ensuite été améliorées jusqu’à leur point culminant : la défaite du joueur d’Echecs Garry Kasparov, vaincu par le super-ordinateur Deep Blue d’IBM.

Plus tard entre les années 1960 et le début des années 1970, les premiers jeux développés durant cette période utilisaient la logique discrète et, le plus souvent, n’intégraient pas d’IA car ils opposaient uniquement deux joueurs.



Figure : Garry Kasparov contre Deep Blue d'IBM

## Les premières IA pour le jeu vidéo

Dans les années 1970, les premiers jeux avec des modes 1 joueur apparaissent. Les plus remarquables sont Chasse au Wumpus et Star Trek en 1972 qui utilisaient des Stored Patterns[[2]](#footnote-3) pour implémenter les IA des ennemis. L’utilisation des microprocesseurs a permis une augmentation du nombre de mouvement.

Dans la fin des années 1970 et jusqu’au début des années 1990 : c’est l’âge d’or des jeux d’arcade. Les IA se popularisent. Nombre de jeux (comme Space Invaders en 1978 et Galaxian en 1979) intégraient maintenant une IA plus complexe avec différents niveaux de difficulté et de nombreux modèles de mouvement.

* **Pacman** (1980) a introduit un pattern IA pour les jeux de labyrinthe, avec des personnalités distinctes pour chaque ennemi.
* **Karate Champ** (1984) a quant à lui introduit le pattern pour les jeux de combat.
* **First Queen** (1988) était un RPG dans lequel des personnages contrôlés par l’IA devaient suivre un leader. Ce type d’IA ayant été amélioré par Dragon Quest IV (1990) avec le system « Tactics » (ajustement des routines utilisées par les IA) puis Secret of Mana (1993).

A partir des années 1990, l’émergence de nouveaux genres de jeux vidéo a conduit à l’utilisation d’outils plus sophistiqués tels que les automates finis. Les stratégies en temps réel requéraient la prise en compte de nombreuses contraintes : de nombreux objets, des informations incomplètes, la recherche de chemin, l’économie du nombre de calculs lors de la planification, etc.

Les premiers jeux rencontraient de nombreux problèmes (Herzog Zwei en 1989 avec une recherche de chemin basée sur un automate fini à trois états, ainsi que Dune 2 en 1992 qui utilisaient de nombreuses « astuces »).

## Une IA évolutive

Longtemps délaissée au profit des graphismes des jeux vidéo, cette branche constitue maintenant une part importante du travail dans un jeu vidéo. Les joueurs réclament de plus en plus des IA au comportement humain afin d'accroître le sentiment d’immersion. Certains jeux sont réputés pour être des précurseurs en matière d’IA, en voici quelques exemples :

- **Creatures** (1998) : ce jeu est célèbre pour être le premier à utiliser l’apprentissage automatique lors d’une simulation interactive. A l’aide des réseaux de neurones, les créatures (appelées « Norms ») apprennent divers comportements. Ils peuvent ainsi interagir avec leur environnement.

- **Halo : Combat Evolved** (2001) : ce jeu utilisait des arbres pour déterminer le comportement des PNJs (« Behavior Tree »), avec beaucoup d’attention portée sur le moindre détail du jeu. Ainsi, la gestion des groupes de PNJs était particulièrement bonne et a joué un rôle de précurseur.

- **F.E.A.R.** (2005) : l’IA utilise un planificateur (« Planner ») afin de générer des comportements sensibles au contexte, ce fut la première fois dans un jeu grand public. On ressent une grande habileté chez les PNJs. Ils sont en effet capables de trouver une couverture derrière des tables, basculer des étagères, ouvrir des portes, passer à travers les fenêtres, etc. Ce jeu constitue une référence en la matière.

- **Black & White** (2001) : le jeu propose au joueur d’incarner une divinité et de guider un peuple. Pour l’aider dans cette tâche, le joueur devra éduquer sa créature, un agent qu’il sera possible de récompenser après une action pour renforcer ce comportement, ou punir, afin de le diminuer.

Une nouvelle perspective pour l’IA

Machine learning avec du Deep Neural Network

IA with procedural content

# Les applications possibles prometteuses

Dans cette section, nous aborderons certaines des techniques les plus utilisées dans les jeux vidéo, plus particulièrement dans le domaine de l’intelligence artificielle. Nous étudierons leurs caractéristiques ainsi que les avantages et les inconvénients de chacune plus particulièrement.

## Le Machine Learning, l’IA qui apprend en même temps que le joueur

### Les réseaux de neurones

Un réseau de neurones artificiel (RNA) est un paradigme de traitement de l’information qui est inspiré de la façon par laquelle le système nerveux biologique, telle que le cerveau, traite l’information. L’élément clé de ce paradigme est la structure novatrice utilisée pour le système de traitement de l’information. Il est composé d’une grande quantité d’éléments de traitement grandement interconnectés (neurones), qui, en travaillant à l’unisson résolvent des problèmes spécifiques. Les RNA, comme les humains, apprennent par l’exemple. Un RNA est configuré pour une application particulière, comme la reconnaissance de motifs ou la classification de données, à travers une procédure d’apprentissage. Apprendre, dans le système biologique, implique des ajustements au niveau des connections synaptiques qui existent entre les neurones. C’est également vrai pour les RNA.

La simulation de réseaux de neurones apparaît comme étant un développement récent. Cependant, ce domaine a été établi avant l’avènement des ordinateurs, et a survécu à au moins un revers majeur et à plusieurs époques. Le premier réseau de neurones artificiel fût produit en 1943 par le neurophysiologiste Warren McCulloch et le logicien Walter Pits. Mais la technologie disponible à cette époque n’a pas permis d’en faire grand-chose.

Les réseaux de neurones, avec leur remarquable capacité de déduire un sens à des données compliquées ou imprécises, peuvent être utilisés pour extraire des motifs ou pour détecter des tendances qui sont trop complexes pour être notifiés par des humains ou bien même d’autres techniques informatiques. Un réseau de neurones entraîné peut être assimilé à un « expert » dans la catégorie d’information qu’il analyse. Cet expert peut être utilisé pour fournir des prévisions en lui donnant de nouvelles situations et répondre à des questions du type « Que se passerait-il si… ? ».

D’autres avantages incluent :

* L’apprentissage adaptatif : Cette capacité permet d’apprendre comme effectuer des tâches suivant les données choisies pour l’entrainement or pour l’expérience initiale.
* L’auto-organisation : Un RNA peut créer sa propre organisation ou représentation de l’information qu’il reçoit pendant la période d’apprentissage.
* Les opérations en temps réel : Les calculs du RNA peuvent être effectués en parallèle, et des équipements spéciaux qui prennent en compte de cette capacité sont en voie de construction.

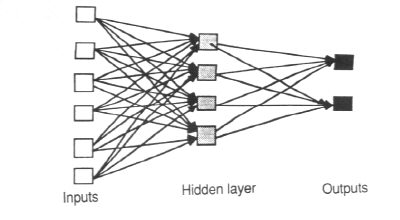


Figure : Un simple réseau de neurones

Le type de réseau de neurones artificiel le plus commun est composé de trois groupes, ou couches, d’unités : une couche d’unités dites « d’entrée » qui est connectée à une couche d’unités « cachées », qui elle-même est connectée à la couche d’unités de « sortie ». (Voir la Figure 1)

* L’activité des unités d’entrée représente l’information brute transmise au réseau.
* L’activité des unités cachées est déterminée par celle des unités d’entrée et des connections pondérées entre les unités d’entrée et celles cachées.
* Le comportement des unités de sortie dépend de l’activité des unités cachées et des liaisons pondérées entre les unités cachées et celles de sortie.

Ce type simpliste de réseau de neurones est intéressant car les unités cachées sont libres de construire leur propre représentation de leurs entrées. Ces poids entre les unités cachées et les unités d’entrée déterminent la valeur d’activité d’une unité cachée, et donc en modifiant ces valeurs, une unité cachée peut choisir ce qu’elle représente.

On peut distinguer les architectures monocouche ou multicouche. Les organisations monocouches, dans lesquelles toutes les unités sont connectées aux autres, constituent le cas le plus général et un plus grand potentiel de puissance de calcul que les organisations multicouches structurées par hiérarchie. Dans les réseaux multicouches, les unités sont numérotées par couche au lieu de suivre une numérotation globale.

Il existe plusieurs stratégies d’apprentissage pour un réseau de neurones artificiel :

* **L’apprentissage supervisé** : Cette stratégie implique un professeur plus intelligent que le réseau lui-même, qui servira de superviseur pour l’apprentissage du réseau de neurones. Par exemple, dans le cas de la reconnaissance faciale, le professeur connaît déjà les noms des personnes qu’il montre au réseau de neurones. Il peut ainsi comparer les réponses données par le réseau et déterminer si des ajustements sont à faire.
* **L’apprentissage non supervisé**: Cette technique est requise lorsqu’il n’existe pas d’exemples d’ensembles de données possédant des réponses connues. Prenons par exemple la recherche d’un motif caché dans un ensemble de données. Une application de cela est le « clustering », c’est-à-dire la division en groupes d’un ensemble d’éléments suivant une disposition inconnue.
* **L’apprentissage par renforcement** : Cette stratégie est basée sur l’observation. Prenons par exemple un petit robot circulant à travers un labyrinthe. Si le robot tourne à gauche, il obtient une récompense, s’il tourne à droite il hérite d’un malus. On peut supposer que le robot apprendra avec le temps à tourner à gauche. Son réseau de neurones prend une décision avec une valeur d’entrée (tourner à gauche ou à droite) et observe son environnement (récompense ou malus). Si l’observation est négative, le réseau peut ajuster les valeurs des poids entre les unités du réseau afin d’effectuer un choix différent la prochaine fois. L’apprentissage par renforcement est une technique commune dans le domaine de la robotique.

Cette capacité qu’a le réseau de neurones à apprendre, à faire des ajustements sur sa propre structure à travers le temps, est ce qui le rend si utile dans le domaine de l’intelligence artificielle.

Afin de mieux comprendre comment fonctionne un réseau de neurones artificiel, regardons plus en détail le fonctionnement interne d’un réseau de neurones, appelé le perceptron. Il a été inventé en 1957 par Frank Rosenblatt au laboratoire aéronautique de l’université Cornell. Il s’agit du plus simple réseau de neurones possible car son modèle de calcul se base sur un seul neurone. Un perceptron est composé de plusieurs points d’entrée, d’un processeur et d’un seul point de sortie.

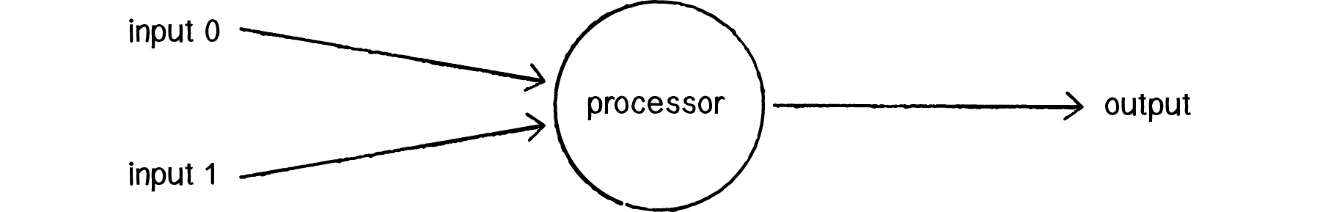


Figure Le perceptron

Un perceptron suit le modèle « feed-forward », ce qui signifie qu’il n’existe aucun cycle au sein du réseau de neurones. Autrement dit les entrées sont envoyés au cœur du neurone qui lui-même transmet ses données à la sortie, comment montré sur le schéma ci-dessus (Figure 4).

Chaque entrée est envoyée au processeur suivant le poids de la liaison qui lui est attribué. Le perceptron calcule ensuite la somme des valeurs qu’il a reçu et à l’aide de sa fonction d’activation choisit quel type de réponse il doit envoyer à l’unité de sortie.

Un perceptron peut bien avoir plusieurs points d’entrée, cependant il reste un neurone seul. Le pouvoir du réseau de neurones réside dans l’interconnexion elle-même. Les perceptrons sont, tristement, plutôt limité dans leur capacités. En lisant un livre sur l’intelligence artificielle on peut y apprendre qu’un perceptron peut seulement résoudre des problèmes dissociables linéairement. Un des exemples les plus simples d’un problème dissociable non-linéairement est le XOR, ou encore « ou exclusif ». Tandis que les opérateurs logiques AND et OR sont des problèmes séparables linéairement, il en est autrement du XOR.

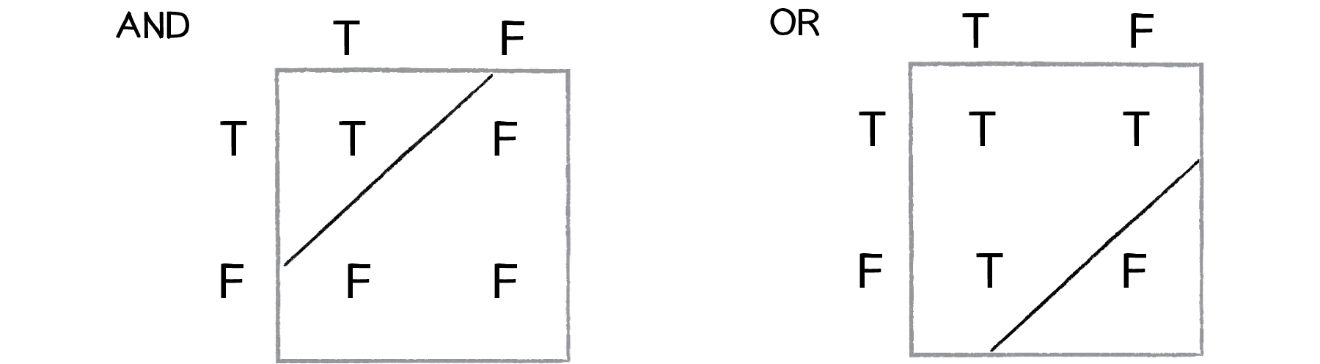


Figure Table de vérité AND et OR

Ici il est possible de tirer une ligne afin de séparer les réponses possibles du perceptron.

Le XOR est l’équivalent du OR et du NOT AND. En d’autres termes A XOR B vaut vrai seulement si un seul des deux vaut vrai.

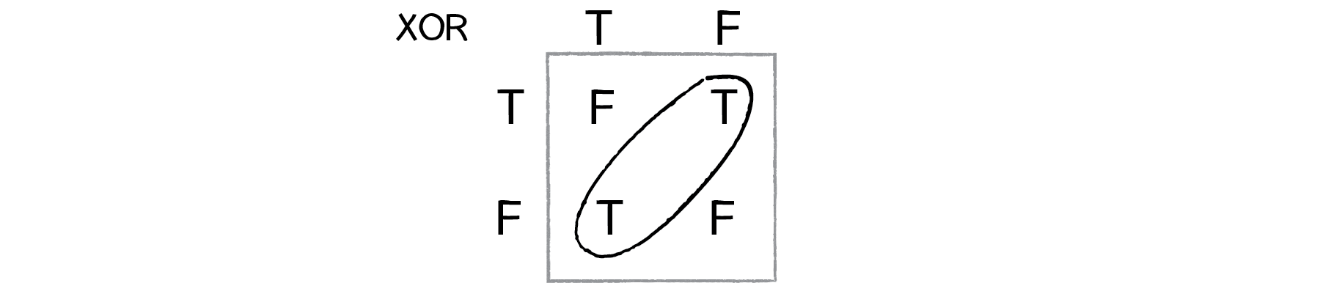


Figure Table de vérité XOR

Le problème représenté ci-dessus n’est pas séparable linéairement. En effet, il est impossible de séparer les deux types de sorties par une simple ligne droite.

Nous venons de voir qu’il était impossible pour un perceptron de résoudre le problème du XOR. Mais il n’est pas impossible pour un réseau de deux perceptrons de résoudre le XOR. Si un perceptron résout le OR et qu’un autre se charge du NOT AND alors les deux perceptrons combinés peuvent venir à bout du XOR.

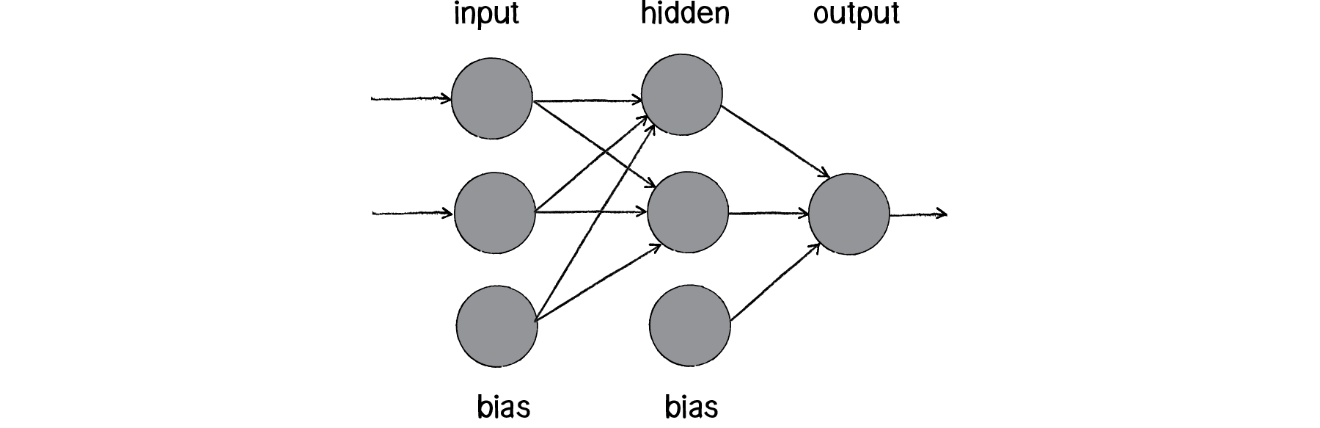


Figure Un perceptron multicouche

Entraîner ce genre de perceptrons (Figure 7) est beaucoup plus compliqué. Avec un simple perceptron, il était facile d’évaluer la manière de changer les poids suivant l’erreur obtenue. Mais ici, il y a tellement de connexions différentes entre les neurones qu’il en devient difficile de déterminer l’influence qu’on les neurones entre eux.

Cependant il existe une solution pour optimiser les poids d’un réseau de neurones multicouches qui est connue sous le nom de rétropropagation du gradient (ou backpropagation en anglais). La sortie du réseau est générée de la même manière qu’un perceptron. La différence ici est que les données passent par une couche supplémentaire de neurones avant d’atteindre la sortie. Entraîner le réseau (c’est-à-dire ajuster les poids) implique aussi de prendre en compte l’erreur (la différence entre le résultat désiré et la supposition). L’erreur, néanmoins doit être retransmise à travers tout le réseau. La dernière erreur ajuste en fin de compte le poids de toutes les connexions.

Actuellement il est courant d’utiliser les réseaux de neurones dans le domaine de l’intelligence artificielle. Mais qu’en est-il de l’intelligence artificielle (IA) dans les jeux vidéo ?

Il existe un logiciel implémentant une méthode de neuro évolution appelé NeuroEvolution of Augmenting Topologies (**NEAT**). Ce logiciel peut faire évoluer des réseaux de neurones d’une complexité sans limite depuis un état de départ. Il est donc possible, à l’aide d’une des implémentations faite dans un langage de programmation (tel que le C++ ou le Java), d’intégrer cette méthode dans un moteur de jeu vidéo.

C’est le cas de **NERO** (Neuro-Evolving Robotic Operatives) qui utilise l’implémentation C++ de la méthode NEAT. Ce jeu vous permet de jouer avec des agents intelligents adaptables par le joueur lui-même. Il vous sera possible d’évoluer votre propre armée de robots en customisant leur intelligence artificielle afin d’accomplir des tâches éprouvantes. Ensuite, invoquez la dans l’arène et défiez celle de vos amis dans des compétitions en ligne.



Combat d’armée dans NERO

### Les arbres de comportement

Dans le domaine des jeux vidéo, il est important de développer des agents dotés d’une intelligence artificielle qui puissent interagir avec l’environnement ou même avec d’autres agents. Dans les jeux un agent peut être un objet, une créature, une personne ou même un système agissant dans le monde virtuel. Suivant le genre, les mécaniques ou le but du jeu, les agents virtuels peuvent être bénéfique pour plusieurs raisons, telle que :

* Fournir des ennemis et des alliés au joueur, et donc créer du contenu au jeu.
* Créer des entités intelligentes qui viennent compléter le monde virtuel, rendant ainsi l’expérience du joueur plus immersive.
* Créer des personnages capables de capter l’empathie du joueur.

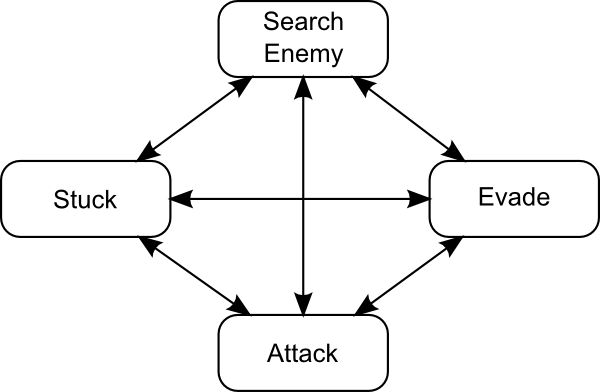
En général les agents virtuels peuvent interagir avec les humains (sous forme d’avatar dans le jeu) ou avec d’autres agents artificiels. Les agents autonomes intelligents dans les jeux sont appelés PNJ (Personnage Non Joueur) et possèdent des organes sensoriels ou des mécanismes d’actions particuliers à leur but mais aussi à l’environnement qui les entoure. Un PNJ peut, par exemple, être un vendeur de potion humain restant figé dans la ville, attendant que le joueur vienne lui cliquer dessus, ou alors incarner le compagnon du joueur principal, le suivant ainsi durant toutes ses aventures.

Pour développer des agents intelligents, il est nécessaire de construire un vaste répertoire de comportements. Un comportement implique le contrôle de plusieurs actions à travers le temps. Par exemple, imaginez un jeu d’action où les PNJ ont un comportement d’auto-protection. Pour accomplir ce comportement, le PNJ doit vérifier s’il est en danger (par exemple, si sa vie est basse), si c’est confirmé, le PNJ devra alors s’éloigner du joueur. Le nombre de comportements qu’un agent peut utiliser indique le nombre de situations différentes qu’il peut reconnaître et dont il peut agir en conséquence.

Il est aussi important que la méthode utiliser pour contrôler les agents permette une coordination efficace des comportements, afin qu’un agent puisse accomplir son but efficacement, c’est-à-dire sans effectuer d’actions aléatoires, inutiles ou stupides (dans un combat, un PNJ ne peut pas s’arrêter pour vendre des objets).

Les agents intelligents possédant des larges répertoires de comportement efficace rendent l’expérience interactive avec l’humain plus intéressante. De plus, dans les jeux vidéo, ces agents ne frustrent pas le joueur, lui permettant une meilleure immersion dans le jeu. Cependant ces résultats ont un coup en terme de complexité tels que :

* **La vitesse** : Beaucoup de comportements implique une forte consommation de la mémoire pour les conserver et une forte utilisation du processeur pour les exécuter. Cependant dans les jeux, rare est la place faite à la partie IA dans la répartition de l’utilisation mémoire/processeur, bien souvent au profit du rendu graphique.
* **La transparence et la fiabilité**: Les développeurs doivent avoir un contrôle total sur les actions de leurs agents. Il ou elle doit être capable de déboguer et de savoir, de manière relativement simple, l’état interne de l’agent, en étant capable de prédire approximativement ses prochaines actions. Ce besoin est particulièrement important pour des raisons de sécurité. Perdre le contrôle de ce que l’agent fait et du moment où il doit le faire peut causer des soucis. En effet, des agents sans contrôle peuvent ruiner l’expérience de jeu. Par conséquent, la quantité de comportements peut rendre le développement plus compliqué et résulter en un système peu fiable.



Une machine à états finis

L’arbre de comportement est venu remplacer la machine à états finis dans le domaine du jeu vidéo en apportant plusieurs avantages :

* **La maintenabilité** : les transitions de l’arbre de décision sont définies dans sa structure, et non par les conditions à l’intérieur des états. Grâce à cela, les nœuds peuvent être conçus indépendamment les uns des autres, ainsi, lorsqu’on ajoute ou retire de nouveaux nœuds (ou même des sous arbres) dans un petite partie de l’arbre, il n’est pas nécessaire de changer les autres parties du modèle.
* **La « scalabilité »**: Lorsqu’un arbre de comportement possède beaucoup de nœuds, il est possible de le décomposer en petits sous arbres, préservant ainsi la lisibilité du modèle graphique.
* **La réutilisation**: Grâce à l’indépendance des nœuds dans l’arbre de comportement, les sous arbres sont eux aussi indépendants. Cela permet la réutilisation des nœuds ou des sous arbres à travers d’autres arbres ou projets.
* **La spécialisation**: Bien que les nœuds de l’arbre soient indépendants, ils restent reliés à cause de la structure de l’arbre. Ceci permet au concepteur de construire des sous arbres spécifiques à une tâche sans perdre la flexibilité de son modèle.
* **La parallélisation** : Les arbres de comportements peuvent spécifier des nœuds parallèles qui exécuteront leurs enfants au même instant sans perdre le contrôle du modèle d’exécution. Cela est possible car la parallélisation est contenue localement au nœud parallèle.

Le modèle est composé de nœuds et d’arêtes. Pour une paire de nœuds connectés par une arête, le nœud du dessus est appelé parent et celui du dessous enfant. Il n’y a aucune limite sur la quantité d’enfants qu’un nœud peut posséder. Les nœuds sans enfants sont appelés feuilles tandis que le nœud sans parent est appelé racine. Un nœud se situant entre la racine et les feuilles peut être de deux types, composite ou décorateur. Chaque sous arbre définit un comportement différent, qui peut être basique (composé de peu de nœuds), ou bien plus complexe (composé d’un grand nombre de nœuds).

La racine de l’arbre génère un signal périodiquement suivant un fréquence f. Le signal se propage à travers les branches de l’arbre suivant l’algorithme défini par chaque type de nœud. Lorsque le signal atteint une feuille, le nœud effectue quelques calculs et retourne une valeur d’état. Ensuite la valeur est renvoyée à travers l’arbre dans le sens inverse. Le processus est terminé lorsqu’une valeur d’état est remontée jusqu’au nœud racine.

## Dynamic Game Difficulty Balancing, l’IA qui s’adapte au niveau du joueur

Beaucoup d’exemples assez simple (on augmente automatiquement le niveau/nombre des ennemis, etc si on trouve que le joueur s’en sort bien, et inversement)

Les fameux items de Mario Kart (plus t’es loin, meilleurs sont tes objets)

Le meilleur exemple pour ce type d’IA se trouve du côté de Valve, et plus précisément dans leur FPS de survival horror, Left 4 Dead. Dans celui-ci, **là on explique brièvement le jeu, on sait jamais**. Et, contrairement à beaucoup de jeux dans lesquels la difficulté augmente de façon constante, ici un “AI Director” analyse constamment les performances individuelles et collectives des joueurs pour construire une expérience adaptée à tous.

Figure 9 : The Director Directs, de Tim Buckley

#### Les avantages

#### Les inconvénients

**Temps de calcul**

Un des principaux freins de l’IA dans les jeux vidéo, c’est l’impact sur les performances. En effet.

**Résultats plus imprévisibles**

L’utilisation de beaucoup d’algorithmes pour faire évoluer puis déterminer le comportement d’un agent à un moment donné fait qu’il est plus difficile de prédire ses actions.

## Des jeux par les IA, pour les IA

Bientôt, il n’y aura plus du tout besoin d’humains dans la boucle de création et consommation de jeu.

En effet, la prochaine IA dont nous allons parler est une IA capable d’apprendre à joueur à des jeux, tandis que la suivante, très jeune, est capable de développer des jeux (simples) complets.

<<< IMAGE >>>

Légende de l’image

### General Game Playing, l’IA qui apprend à jouer

C’est un domaine qui a gagné en popularité ces dernières années, notamment grâce à une IA qui a gagné le prix du public de la compétition annuelle organisée par l’AAAI (Association for the Advancement of Articifial Intelligence).

Cette IA est capable d’apprendre à jouer à un niveau de Mario Bros (un jeu de plateforme 2D) et de s’adapter aux différents niveaux qui le composent.

#### Les avantages

#### Les inconvénients

**Temps de calcul**

Un des principaux freins de l’IA dans les jeux vidéo, c’est l’impact sur les performances. En effet.

**Résultats plus imprévisibles**

L’utilisation de beaucoup d’algorithmes pour faire évoluer puis déterminer le comportement d’un agent à un moment donné fait qu’il est plus difficile de prédire ses actions.

### Angelina, l’IA qui développe des jeux

Angelina est une IA qui, à partir d’une description en langage naturel, peut générer des jeux de plateforme 2D, avec une ambiance, des graphismes et un gameplay qui respectent (dans une certaine mesure) la description fournie.

C’est un sujet de recherche développé par Michael Cook, un chercheur britannique à l’université de Falmouth.

Figure : Exemple de jeu produit par Angelina

Pour arriver à ce résultat, elle analyse la description, **<<<DESCRIPTION>>>**

#### Les avantages

**Une interaction avec l’environnement plus réaliste**

Ouais toussa toussa

**Une autonomie et une adaptabilité accrues**

En ce moment, la mode est au procédural. Or, il est très difficile, sinon impossible de prédire tous les cas de figures et donc de coder les comportements. Il suffit donc d’apprendre à un agent d’apprendre par lui-même pour lui permettre de se débrouiller

Détournement intelligent de “Donne un poisson à un homme, il mangera un jour. Apprend lui à pêcher et il mangera toute sa vie”, mais pour l’IA.

Teach an AI to XXX, it’ll XXX. Teach it to learn and it’ll take over the world.

**Un gameplay plus vivant**

C’est quand même mieux qu’un gameplay mort.

C’est tout pour moi, merci.

**Une rejouabilité accrue**

Comme avec le procédural, ça augmente la durée de vie et la rejouabilité.

#### Les inconvénients

**Temps de calcul**

Un des principaux freins de l’IA dans les jeux vidéo, c’est l’impact sur les performances. En effet.

**Résultats plus imprévisibles**

L’utilisation de beaucoup d’algorithmes pour faire évoluer puis déterminer le comportement d’un agent à un moment donné fait qu’il est plus difficile de prédire ses actions.

# De l’importance de l’IA pour l’immersion du joueur

Dans cette partie, nous parlerons de l’intelligence artificielle qui n’est pas flagrante, mais qui rend, inconsciemment pour le joueur, l'expérience de jeu plus réaliste et immersive. Cette intelligence artificielle est principalement importante dans les jeux à un seul joueur, dans lesquels l’immersion est bien plus importante que dans les jeux multi-joueurs, dans lesquels il est dur de maintenir le joueur dans un univers fictif à cause de la présence d’autres joueurs dont le comportement est difficilement contrôlable et peut être perturbateur.



Figure : Deadpool, personnage de l’univers Marvel connu pour ne pas beaucoup respecter le 4ème mur

## Suspension consentie de l'incrédulité, solipsisme et vallée dérangeante

Lorsqu’un PNJ répète sans cesse le même dialogue au joueur, celui-ci est ramené à la réalité. Quand un PNJ rencontre un problème de recherche de chemin (lorsqu’un PNJ fonce obstinément dans un mur) ou n’arrive pas à se déplacer d’un point à un autre, le joueur est ramené brutalement à la réalité.

Figure 12 : Why RPG NPCs Always Repeat Themselves, de Dorkly

Toutes ces actions, souvent sans le vouloir, cassent le quatrième mur, et mettent fin à ce que l’on appelle la **Suspension Consentie de l’Incrédulité** (aussi connue sous le nom de **Suspension of disbelief** en anglais). Il s’agit d’une sorte de contrat que le joueur signe avec l’œuvre de fiction (que ce soit un jeu, un film, un livre), dans lequel le joueur accepte de mettre de côté son scepticisme le temps de la consultation de l’œuvre en question.

Il peut ainsi s’immerger dans un univers avec des éléments imaginaires, pour peu que ceux-ci restent crédibles, et que les règles qui régissent l’œuvre restent constantes.

Lorsque ce contrat est rompu (par définition, il ne peut être rompu que par l’œuvre de fiction), il est difficile de faire replonger un joueur dans l’univers du jeu, avec lequel il restera toujours un peu distant, et cela renforce le solipsisme du joueur.



Figure 13 : Why RPG NPCs Always Repeat Themselves, de Dorkly

### Solipsisme

Le solipsisme (du latin *solus*, « seul » et *ipse*, « soi-même ») est une attitude selon laquelle, la seule certitude absolue pour le sujet pensant, est que lui-même existe. Cette attitude est répandue dans la société, et il est tout à fait normal de l’avoir ressenti de façon passagère.

Mais c'est dans les jeux vidéo (principalement ceux à un seul joueur) qu'elle est le plus présente, et qu’on essaie le plus de la faire disparaître. En effet, le but d’un jeu narratif est d’immerger le joueur, et un très bon moyen pour cela est de lui faire oublier qu’il est la seule personne effectivement douée de conscience.

Figure 14 : Why RPG NPCs Always Repeat Themselves, de Dorkly

Evoluer dans un monde photoréaliste et crédible est un bon moyen de garder le joueur immergé. Toutefois, rares sont les jeux sans PNJ, il faut donc s’assurer qu’eux aussi ne cassent pas cette immersion. Pour cela, les doter de réactions naturelles est la manière la plus efficace pour arriver à ce résultat.

Ainsi, de de plus en plus de développeurs de MMORPG (Jeu de Rôle Massivement Multijoueur) décident de donner plus de profondeur aux PNJ (amicaux comme ennemis).

C’est le cas notamment dans EverQuest Next, dans lequel chaque PNJ se voit attribuer certains traits de caractère, ce qui affecte la façon dont il interagit avec les joueurs, ainsi qu’avec les autres PNJ.

De plus, toujours dans EverQuest Next, les monstres ne sont pas regroupés en « camps de monstres », dans lesquels ils ne font qu’attendre, immobiles et apathiques, qu’un joueur vienne les sortir de leur léthargie, comme c’est le cas dans beaucoup de MMORPG actuels.

Par exemple, un monstre pourra former un groupe avec d’autres monstres et tendre des embuscades sur une route, et soit se faire chasser par des joueurs, soit partir vers d’autres horizons s’ils jugent que cette zone n’est pas assez fructueuse.

Toutefois, si le réalisme du comportement des PNJ dépasse un certain point, mais n’arrive pas à la hauteur des attentes du joueur, celui-ci peut être confronté à la **vallée dérangeante** (plus connue sous son nom anglais **d’Uncanny Valley**).



Figure 15 : Why RPG NPCs Always Repeat Themselves, de Dorkly

### Vallée Dérangeante

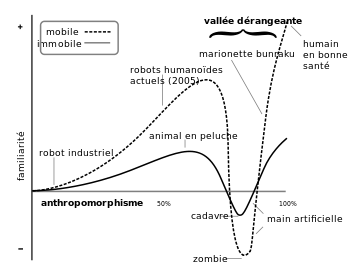
La vallée dérangeante est une théorie selon laquelle plus une créature à forme humaine ressemble à un être humain, plus on remarque ses imperfections, même minimes, et plus elles nous dérangent. Cette théorie a été publié en 1970 par le roboticien Japonais Masahiro Mori, dans le cadre de ses recherches sur les réponses émotionnelles des robots.

Figure : Réaction émotionnelle supposée de sujets humains à l'anthropomorphisme

(Wikipédia)

Dans les jeux vidéo, c’est le plus souvent dû au physique (proportions du corps) et à l’animation (principalement du visage) des personnages humanoïdes, qui se rapprochent de plus en plus de la perfection, et qui donc, à la moindre imperfection, nous paraissent monstrueux.



Figure 17 : Commandant Shepard et son sourire dérangeant, Mass Effect 3, BioWare

Toutefois, le comportement des PNJ joue également un rôle très important. Le personnage de Natalya, personnage secondaire de GoldenEye 007 (Nintendo 64, 1997) notamment, est devenu célèbre pour ses piètres capacités intellectuelles. En effet, plusieurs missions le long du jeu requièrent de l’escorter d’un endroit à un autre en la gardant en vie. Hors, en présence d’ennemis, Natalya n’hésitera pas à se jeter fièrement dans la bataille, bien qu’elle ne puisse ni attaquer ni se défendre, ce qui dans la plupart des cas la mènera à sa mort, et donc à l’échec de la mission.

Figure : Natalya, regrettant sa décision de s’être jetée au milieu de PNJ hostiles

Lorsque ce n’est pas un être humanoïde qu’il voit agir de façon si incohérente, le joueur peut imaginer des causes, ou même ne pas vraiment comprendre mais se dire qu’il lui manque les clés pour comprendre le comportement du PNJ si le joueur ne peut pas s’identifier à lui.

Mais lorsqu’un humain voit un autre être qui lui ressemble, il ne peut s’empêcher d’essayer de se mettre à la place de l’autre et de le comprendre. Ainsi, pour un joueur, être témoin d’un comportement étrange et inexplicable d’un PNJ a forme humanoïde est une expérience parfois drôle, mais souvent désagréable, et qui nuit toujours à l’immersion.

## Entre la sérendipité et l’apophénie

De par son côté de plus en plus évolutif et complexe, il est parfois difficile de prévoir tous les résultats possibles d’une interaction du joueur avec l’IA, ce qui donne parfois lieu à une sorte de gameplay émergent, avec des PNJ au comportement imprévisible, même pour les créateurs.

Si dans certains jeux, cet aspect peut presque passer inaperçu (Fable), d’autres jeux reposent essentiellement sur ce principe (Evie, Cleverbot).

### Fable, un monde plus vivant et cohérent qu’il n’y paraît

Fable est un jeu sorti en 2004, et innovant pour son époque, de par la liberté qu’il laisse au joueur. En effet, le joueur peut se marier, divorcer, boire, jouer aux cartes, acheter, louer et revendre une ou plusieurs maisons, se tourner vers les forces obscures ou le bien, massacrer des innocents, des gardes ou des bandits, entre autres activités diverses et variées.

Ce qui suit est une anecdote tirée du blog d’un des développeurs, traduite, adaptée et commentée.

Les développeurs se sont également imposés deux règles qu’ils jugeaient vitales que le jeu suive, sans quoi il ne respecterait pas leur vision :

* Le personnage principal doit exprimer visuellement ce qu’il ressent.
* Le monde dans lequel le joueur évolue, ainsi que tous ses occupants doivent réagir de manière appropriée aux actions du joueur.

**Un monde simulé**

Pour rendre le monde plus intéressant, les développeurs ont donné à chaque PNJ leurs propres traits de caractères, comme dans EverQuest Next (PC, en développement).

Ils ont également remarqué que les joueurs étaient particulièrement sensibles aux émotions des enfants. Ils ont donc décidé de rendre le comportement des enfants très varié. Ainsi, certains sont de véritables petites pestes (comme certains le sont dans la réalité), et poursuivent et tourmentent les poules, tandis que d’autres vont à l’école bien en rangs, et rentrent gentiment à la maison à l’heure pour le goûter (comme certains le font dans la réalité).

Figure : Enfant tourmentant une poule dans Fable

Et, peut-être le plus important, les enfants fixeront avec intensité ce qu’ils trouvent intriguant.

**Un héros expressif**

Plutôt que d’afficher plein de caractéristiques comme dans un RPG classique, les développeurs ont trouvé plus intéressant de montrer par des mouvements et des expressions que le personnage réagit au monde qui l’entoure (plutôt que de rester planter là, sans émotions apparentes).

**Un héros expressif dans un monde simulé**

Quand on mélange les deux, voici ce que cela peut donner :

Imaginez-vous la scène, une radieuse journée à Bowerstone (une ville du jeu). Les papillons papillonnent, le parfum des tulipes vous est porté par une brise. Au loin, on entend des enfants jouant et rigolant dans une cour de récréation, réticent à retourner en classe, malgré le son de cloche qui signifie la fin de la pause.

Le cours a repris, après quelques remontrances de l’institutrice. Un des enfants, trouvant le cours assommant, regarde par la fenêtre en soupirant.

L’homme s’infiltre silencieusement dans l’école. L’institutrice lui tourne le dos. Elle ne se doute de rien.

Un par un, les enfants se mettent à fixer quelque chose derrière l’institutrice. La classe devient silencieuse. Finalement, l’institutrice entend un bruit derrière elle, se retourne et …

… tombe nez à nez avec un homme en caleçon, planté au milieu de sa salle de classe, souriant à pleines dents. Soudain, souriant de plus belle, il fait un doigt d’honneur. Tout le monde hurle …

**Ce qu’il s’est vraiment passé**

* Le personnage sourit car il se sent sécurité
* Les enfants fixent ce qui leur semble intéressant (ici, le personnage, qui est une nouveauté pour eux)
* Le doigt d’honneur est considéré comme un acte d’agression, d’où les cris

On pourrait ici presque parler de « comportement émergent » (en parallèle au gameplay émergent, dans lequel les joueurs trouvent de nouvelles façons de jouer que les développeurs n’avaient pas prévu) car en partant de règles simples, on arrive à une situation complexe et crédible.

C’est un bon exemple d’apophénie (fait de donner un sens particulier à des évènements banals), car, le joueur interprète les réactions des PNJ différemment de ce que les développeurs avaient probablement en tête lors de la réalisation du jeu.

### De Jabberwacky à Evie, en passant par Cleverbot

Un agent conversationnel (ou chatbot en anglais), est un logiciel autonome qui a pour fonction de dialoguer avec l’utilisateur. Pour cela, la phrase entrée par l’utilisateur est traduite en un langage intermédiaire pour permettre à l’agent de l’analyser, qui va ensuite générer une réponse plus ou moins approprié.

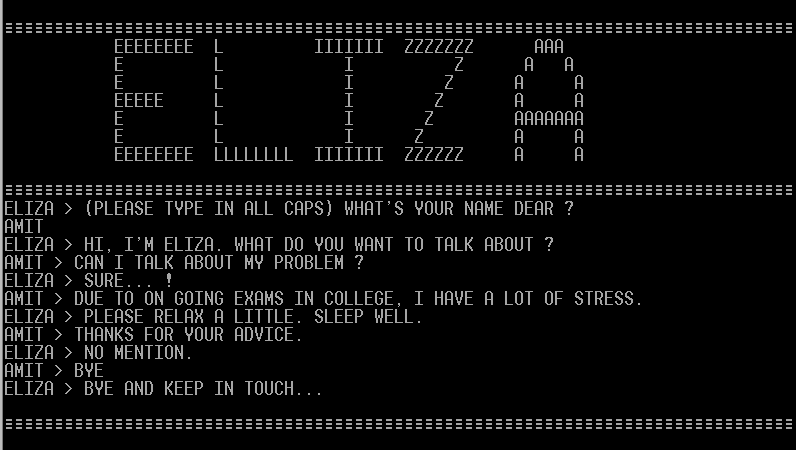
ELIZA, créé en 1966, par Joseph Weizenbaum, MIT, fut le premier agent conversationnel développé. Son comportement était très simple et scripté (elle répondait aux questions en demandant pourquoi cette question, si elle détectait le mot « ordinateur », elle demandait si la question lui avait posée parce qu’elle est une machine), et laissait donc peu de place à une vraie conversation.

Figure : ELIZA, premier chatbot

En 1997, Rollo Carpenter, un informaticien britannique, crée Jabberwacky. C’est lui aussi un agent conversationnel, mais au lieu d’avoir un comportement scripté, celui-ci est capable d’apprendre. En effet, il stocke dans une base de données toutes les conversations des utilisateurs, et s’en sert pour choisir une réponse à la question posée par l’utilisateur.

Ce n’est que onze ans plus tard, en 2008 que Rollo Carpenter met en ligne Cleverbot, descendant spirituel de Jabberwacky. Celui-ci dispose également d’une base de données des conversations avec les utilisateurs parmi lesquelles il va chercher la réponse la plus appropriée, mais a également une notion de contexte, grâce à un système de mémoire à court terme, qui donne un poids plus important aux conversations récentes.

C'est également à ce moment que Existor voit le jour, fondé par Rollo Carpenter, entreprise qui se donne pour objectif de développer des technologies pour arriver à un niveau de conversation humain.

Ainsi, c’est en 2015 que Evie, la petite sœur de Cleverbot, voit le jour. Devant la quantité de données que les différents chatbot de Existor récoltent chaque jour (5.5 millions de conversations par jour), et le nombre impressionnant de 1.5 milliards de conversations enregistrées, les développeurs de Existor ont commencé à utiliser des techniques de Machine Learning.

Dans un premier temps, ils s’en sont servi pour chercher des motifs récurrents dans les phrases, puis, à l’aide de RNNLMs (Recurrent Neural Network Language Models), ils ont pu estimer les probabilités que tels mots se suivent, afin de former une phrase plus naturelle.



Figure : Evie

# Conclusion

Longtemps en retrait dans les jeux vidéo

# Références

* Auteur(s)/Site. nom de la référence, date.
* Auteur(s)/Site. nom de la référence, date.
* Auteur(s)/Site. nom de la référence, date.
* Auteur(s)/Site. nom de la référence, date.
* Auteur(s)/Site. nom de la référence, date.
* Auteur(s)/Site. nom de la référence, date.

1. Le Ferranti Mark I fut le premier ordinateur électronique généraliste commercialisé du monde.

   En novembre 1951, Dietrich Prinz écrivit un des premiers jeux vidéo, un programme d’échecs, pour le Ferranti [↑](#footnote-ref-2)
2. Les stored patterns sont des suites d’instructions répétées en boucle par l’agent. Hormis ce qui lui est dit de faire il ne fera rien d’autre [↑](#footnote-ref-3)