Initiation à l'Intelligence Artificielle

Type d'atelier :

Centre technologique.

Langage / Framework

Cette initiation est réalisée en Lua avec Love2D afin de permettre une approche la plus indépendante possible du moteur de jeu utilisé.

Les concepts abordés sont universels et les algorithmes utilisés sont génériques (non spécifiques au langage Lua).

Cet atelier est donc adapté même si vous travaillez en C#, C++ ou JS.

Introduction

L'IA ! Ca fait rêver !

C'est pourtant assez simple à mettre en place. De plus, pas la peine de viser très haut pour faire de l'IA. Les déplacements des fantômes dans Pac Man c'est déjà de l'IA ! Le vaisseau qui vous tire dessus (voir l'atelier Shooter de Gamecodeur) c'est déjà de l'IA !

Qu'est-ce qui se cache derrière ce terme qui semble tout droit sorti d'un film de science fiction ? Qu'est-ce qu'un agent ? Qu'est-ce qu'une machine à état ? Mon jeu vidéo va t'il prendre le contrôle de l'humanité comme dans Terminator ?

Dans cette atelier d'initiation, je vais donner toutes les bases pour comprendre ce qu'est l'IA (et dédramatiser !) et comment mettre en place de l'IA dans vos premiers jeux vidéo. Nous allons pour cela aborder un peu de théorie, et passer rapidement à la pratique en programmant le comportement d'un robot dans une map.

La théorie

Dans l'univers théorique de l'IA on utilise des termes spécifiques, et notamment la notion d'agent.

C'est un terme un peu pompeux, mais c'est un terme qui a du sens. Vous devez le connaître et le comprendre.

Un agent est une **entité** qui **perçoit** à travers des **capteurs** et qui **réagit** en fonction de **règles** (conditions -> actions).

Retenez les concepts en gras : entité / perception / capteur / réaction / règles.

L'autre concept utilisé universellement dans l'IA, c'est la notion de **machine à état**. Une machine à état permet de décrire ou de programmer un comportement **séquentiel**.

Tout ça vous semble un peu abstrait ? Normal, nous sommes dans un univers théorique qui tend justement à rendre les choses abstraites.

Pas de panique. Relisez 10 fois ma définition il en restera bien quelque chose dans votre esprit. Essayez de comprendre au moins les termes utilisés, ça va vous être utile tout de suite…

Traduction de la théorie en langage de codeur

Maintenant, pour vous aider à comprendre tous ces termes, **on va imaginer programmer un agent** !

De cette manière je vais vous aider à comprendre plus concrètement ce qu'est un agent en passant directement à des notions pratiques !

Imaginez un jeu en vue de dessus, avec une map simple (voir les ateliers Tilemap de Gamecodeur).

1) Remplacez dans votre tête la notion d'**agent** par celle de **personnage**. Dans un jeu vidéo cela peut être un ennemi, un vaisseau, ou encore un robot. **Choisissons un robot !** Ce sont donc les réactions d'un robot, dans un jeu vidéo, que nous allons

programmer.

2) Remplacez dans votre tête la notion de **capteurs** par celle de **fonctions** ou de **variables** que vous pourrez utiliser pour obtenir des informations sur **l'environnement du robot**. Par exemple il pourra savoir ce qui se trouve sur les cases de la map qui l'entourent. Ou encore la position de l'ennemi le plus proche de lui. Mais aussi son énergie, sa propre position, etc. Tout ceci correspond aux capteurs.

3) Replacez enfin dans votre tête la notion de **réaction** par celle d'un **code** qui va décider ce que fait le robot en fonction de son environnement : avancer, attendre, tirer. Et remplacez la notion de de **règles** par celle d'**algorithme (**composé de if ... then … else …). Du genre : si je vois un ennemi, je passe en mode "attaque". Si je ne vois rien, je passe en mode "exploration", etc. Si je n'ai plus assez d'énergie, je me dirige vers la source d'énergie la plus proche. Etc.

C'est plus clair non ? Bon et bien vous avez déjà commencé, dans votre tête à coder un "agent intelligent" et donc à faire de l'IA !

De même, l'idée de "changer de mode" (exploration, attaque, …), on appelle ça une **machine à état**. Encore un mot un peu savant qui cache un concept assez simple comme vous le voyez.

Programmer une machine à état

On va travailler sur la notion de machine à état. Et comme c'est une initiation, on va essayer de voire ça comme quelque chose de très simple.

Le principe d'une machine à état, dans votre jeu vidéo, est de définir les différents états de votre agent (un personnage donc) et les transitions entre ces états. On parle aussi d'automate fini, ou de "machine à état fini". Pourquoi ? Parce qu'on connaît à l'avance tous les états que peut avoir votre agent.

Ensuite la machine va définir quelles conditions font passer d'un état à un autre.

Pour notre zombie par exemple, on va d'abord lister tous les états possibles. J'ai choisi ceux-ci :

- Le zombie rôde (il ère dans une direction aléatoire)
- Le zombie attaque (il va vers un humain)
- Le zombie mord (il inflige des dégâts à l'humain, notamment en mangeant sa chair !)

Et dans le cas d'un comportement collectif :

- Le zombie se dirige vers un autre zombi qui a repéré un humain !

Dans le cas d'un comportement collectif, on parle de "multi-agent".

Maintenant, pourquoi et comment le zombie passe d'un état à un autre ?

Tout simplement, rappelez-vous, en réponse à ce qu'il va observer dans son environnement !

Pour chacun des états possibles, vous devez vous demander ce qui peut se passer.

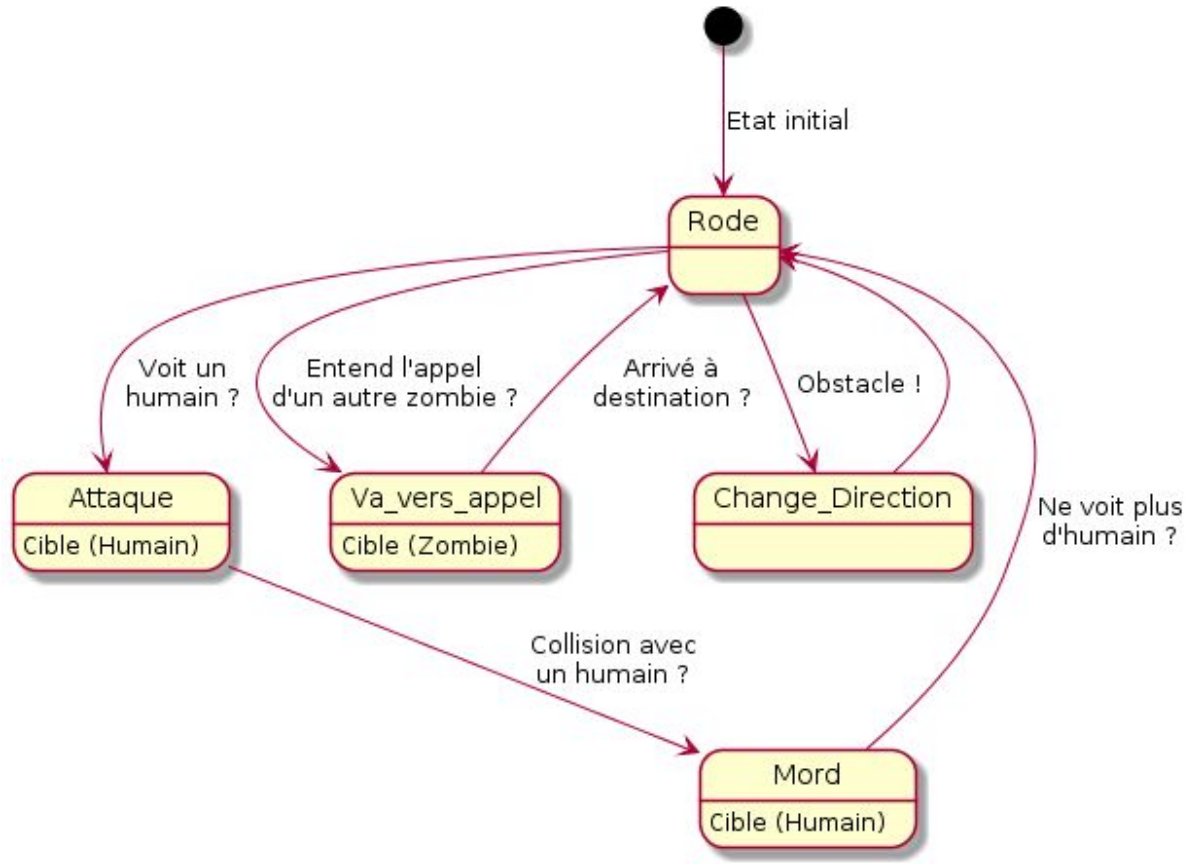
Exemple :

- Quand le zombie rôde, il peut regarder autour de lui, et si il voit un humain à une certaine distance, il va changer d'état et attaquer !
- Quand il attaque, il peut sentir si il est en contact avec sa proie (collision) et il va alors passer à table et mordre !
- Quand il attaque ou mord, il doit bien sûr vérifier que sa cible est toujours là. Elle peut en effet avoir été totalement mangée, ou avoir pris la fuite. Dans ce cas, il va recommencer à rôder.
- Quand il a entendu l'appel d'un autre zombie et qu'il l'a rejoins, le plus simple et de le refaire roder, afin qu'il trouve tout seul l'humain à proximité si il est toujours là !
- Etc.

Entraînez-vous à faire une liste un peu romancée comme celle-ci quand vous élaborez vos états. L'idéal étant une représentation graphique !

Pour représenter cela graphiquement j'ai utilisé un logiciel en ligne (plantuml).

Voici le résultat graphiquement :



[Lien du diagramme]

Casser la rigidité de l'IA

Comme dans la vie, pour qu'un comportement soit réaliste, il doit y avoir un peu de variation, de chaos… Et ça, c'est à vous de le programmer.

Le plus simple est déjà d'utiliser un peu de hasard dans vos choix.

Exemples :

- Quand un Zombie rôde, il est à l'écoute des autres zombies à proximité pour savoir si l'un d'eux a repéré une proie. Mais plusieurs zombies peuvent avoir trouvé la même proie ! On peut alors par exemple lui faire choisir aléatoirement vers lequel de ses camarades il va se diriger au lieu de choisir le plus proche.
- Si il y a plusieurs humains, on peut imaginer que parfois un zombie ne vas pas se diriger vers le plus proche, mais en choisir un qui lui plait ! Soit en choisissant au hasard parmis ceux présents dans un rayon.
- Encore plus fou : quand vous générez un zombie, vous lui donnez des préférences (homme, femme, enfant…) et certains zombies seront donc plus enclins à attaquer des hommes par exemple !

Autres techniques pour ajouter du réalisme :

- Chaque zombie à sa propre vitesse
- Aléatoirement, pendant qu'il rôde, le zombie peut changer de direction
- Aléatoirement, pendant qu'il attaque, le zombie peut changer de vitesse, ou abandonner sa poursuite pendant quelques secondes
- Etc.

Tout cela va rajouter du fun et il n'y a aucune limite à votre imagination !

Références

http://aima.cs.berkeley.edu/