Intelligences Artificielles pour Jeux Vidéo

Intelligence Artificielle

L'intelligence artificielle...

... n'existe pas !

Intelligence Artificielle

L'intelligence artificielle au sens informatique est un ensemble de concepts et de technologies **permettant de résoudre** des problèmes de **complexité forte**.

Intelligence Artificielle

Quelques problèmes relevant de l'IA dans les jeux vidéo :

- Calculs de chemins (Pac-Man),
- Comportements autonomes de PNJ (Sims, Diablo),
- Anticipation de situations (échecs, go, ...),
- Evaluations statistiques et stratégies (RTS, 3X),
- Procédural, création de situations (L4D),
- Aides au game-feel (parkour),
- Animations (Assassin's Creed).

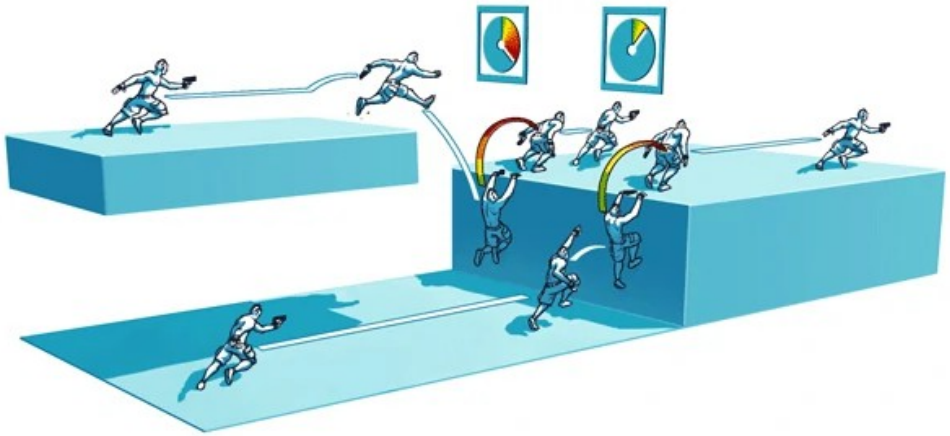
Intelligence Artificielle

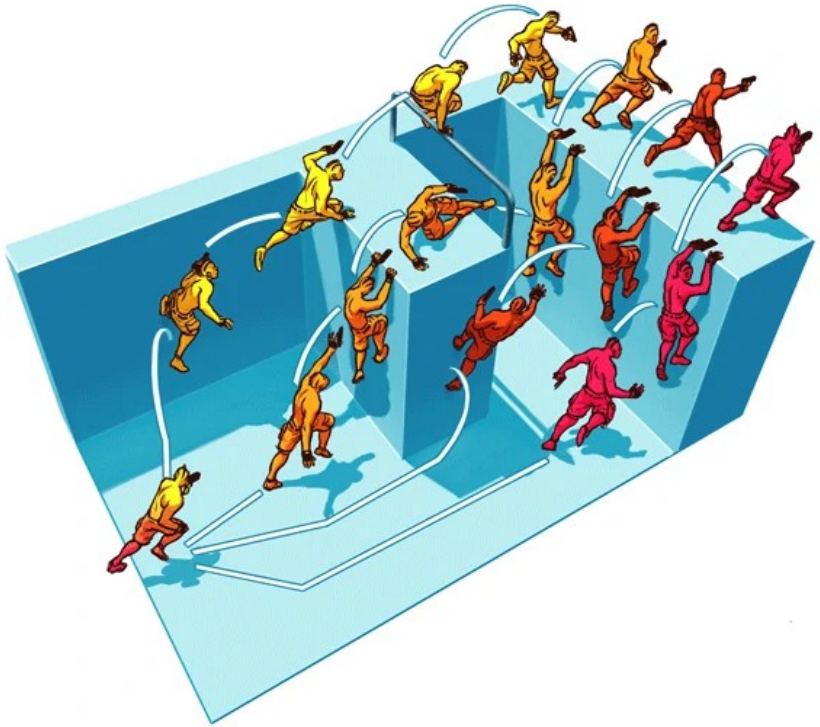
Left4Dead (AI Director)
Les Sims (Smart Objects)

Brink (SMART Freerunning)
AlphaGo (Machine learning)

Left4Dead "AI Director"

Brink "Freerunning"





Les sims SMART Object

Pas d'IA directe sur un sims.

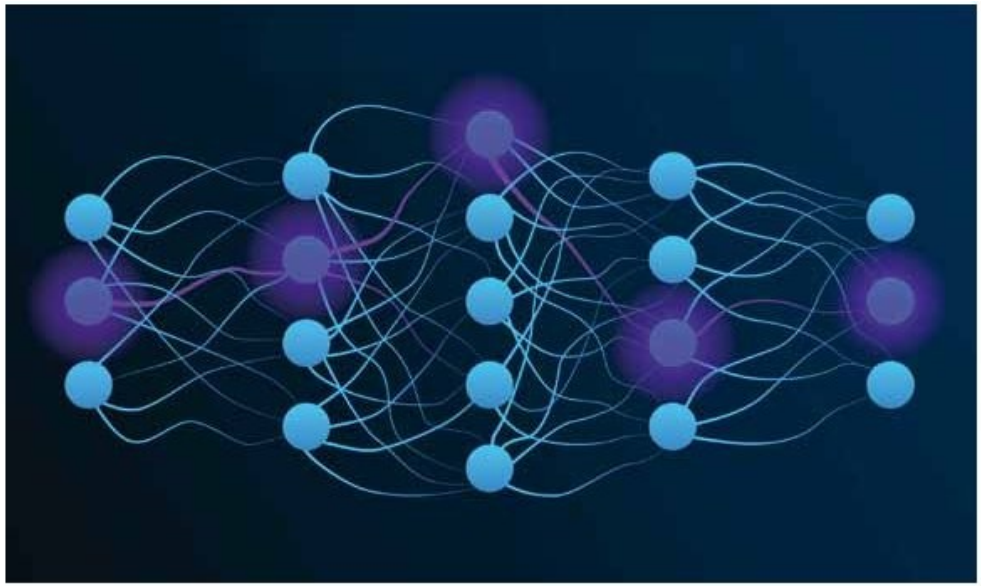
L'IA est donnée par les objets. Pour un objet :

- Visuel + Animations,
- Etat,
- Liste d'actions,
- Comportements sims associés.

Example : the fridge

- Go to a counter
- Prepare the food
- Go to the stove
- Cook the food
- Go to the table (+ chair)
- Eat the food
- Go to the dishwasher
- Clean your plate

AlphaGo



Systèmes d'IA

Systèmes d'IA

Exemples de catégories de problèmes d'IA :

- Perception (i.e. reconnaissance de formes)
- Compréhension (i.e. analyse de langage naturel)
- Décision (i.e. comportement de NPC)
- ...

Un **système d'IA** propose une méthode pour répondre à une catégorie de problèmes.

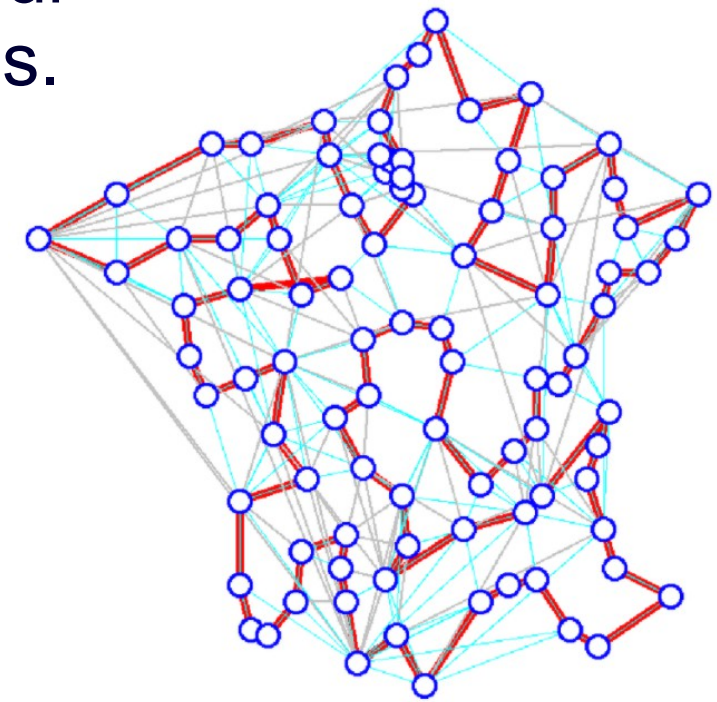
Complexité des problèmes

Les systèmes d'IA ne cherchent souvent pas à résoudre **parfaitement** mais **efficacement** un problème donné.

Complexité des problèmes

Problème du voyageur de commerce :

- Calculer le plus court chemin pour passer une seule fois par N villes.



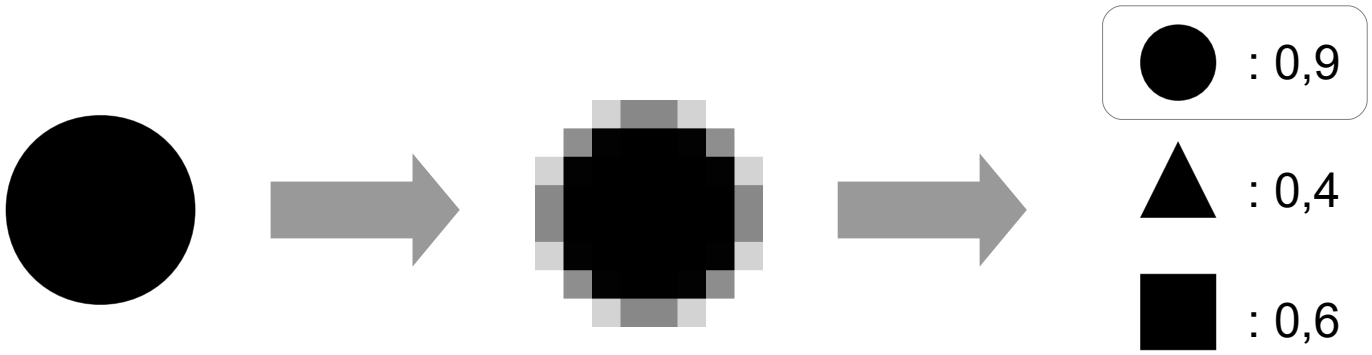
- Recherche exhaustive **naïve** :
  - 10 villes : ~0,2 secondes
  - 15 villes : ~12 heures
  - **20 villes : ~2000 ans !!!**

Problème dit "NP-complet" de complexité en 0(n!)

Heuristique

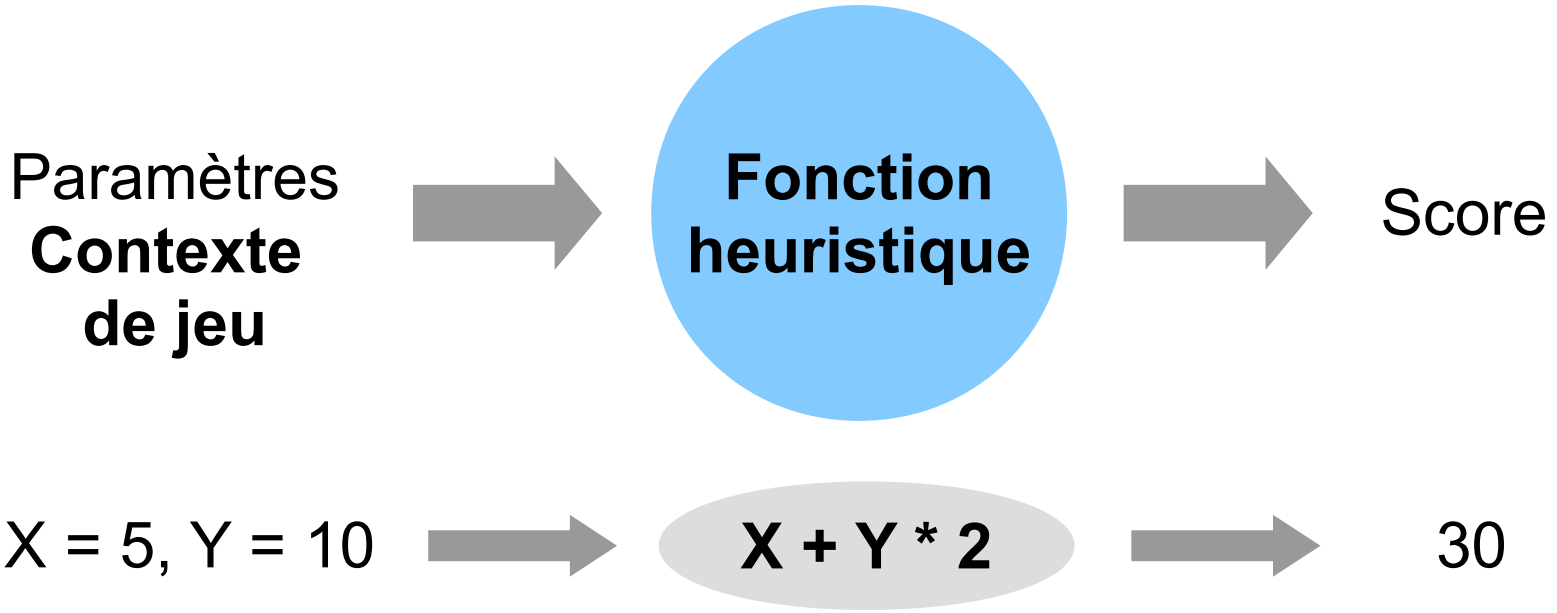
Pour résoudre un problème, un système d'IA :

- Reçoit en entrée un jeu de donnée,
- Transforme en données exploitables,
- Calcule un résultat.



Heuristique

Une **fonction heuristique** sert à évaluer un score à partir de données :



MiniMax

MiniMax

Système de décision MiniMax :

- Simple
- Adapté au "tours par tours".
- Exploration d'arbre de possibilités.

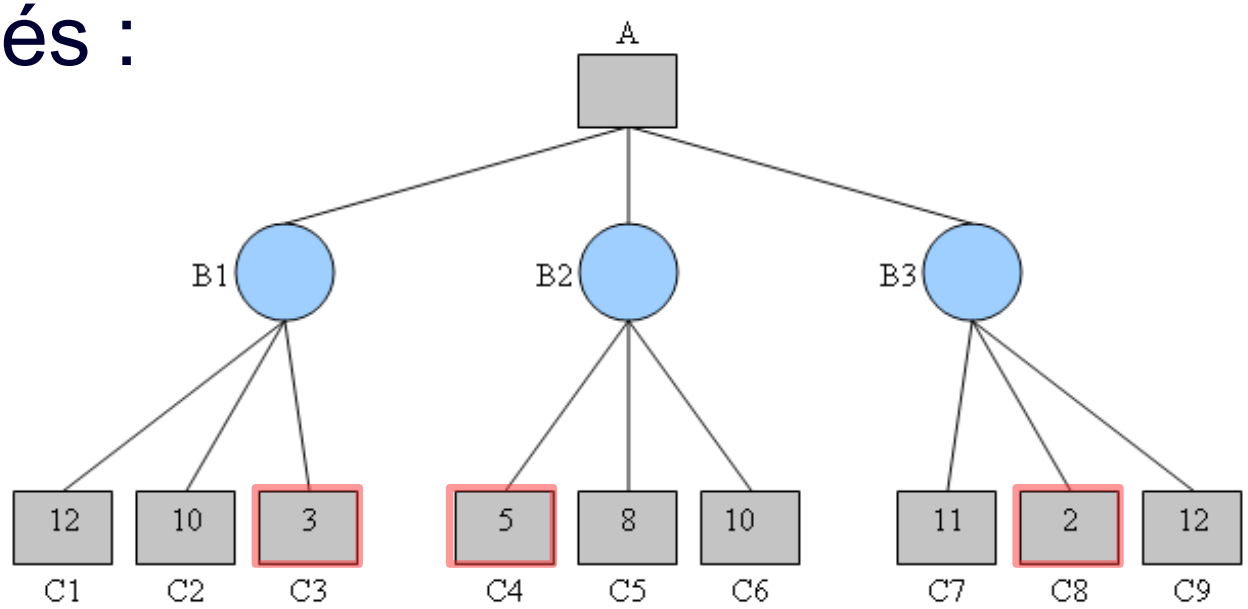
L'IA joue de manière à maximiser les gains en considérant que son adversaire jouera toujours les meilleurs coups.

MiniMax

L'algorithme déroule les coups possibles du joueur et de l'adversaires dans un arbre de possibilités :

Joué par l'IA

Joué par l'adversaire



Ici, B2 est le coup de l'IA qui propose le moins de risque : 5.

MiniMax

Profondeur : Nombre de tours explorés.

MiniMax trouve sa limite dans le nombre de branches à explorer en fonction du nombre de coups possibles (i.e. jeu de go).

MiniMax est donc adapté aux **systèmes simples** avec un nombre d'**actions limité**.

Machine learning

Machine Learning

Le **Machine Learning** a pour but de déterminer une solution à un problème **sans utiliser d'instructions explicites** sur la manière de résoudre ce dernier.

Machine learning

Plusieurs catégories d'algorithmes de ML :

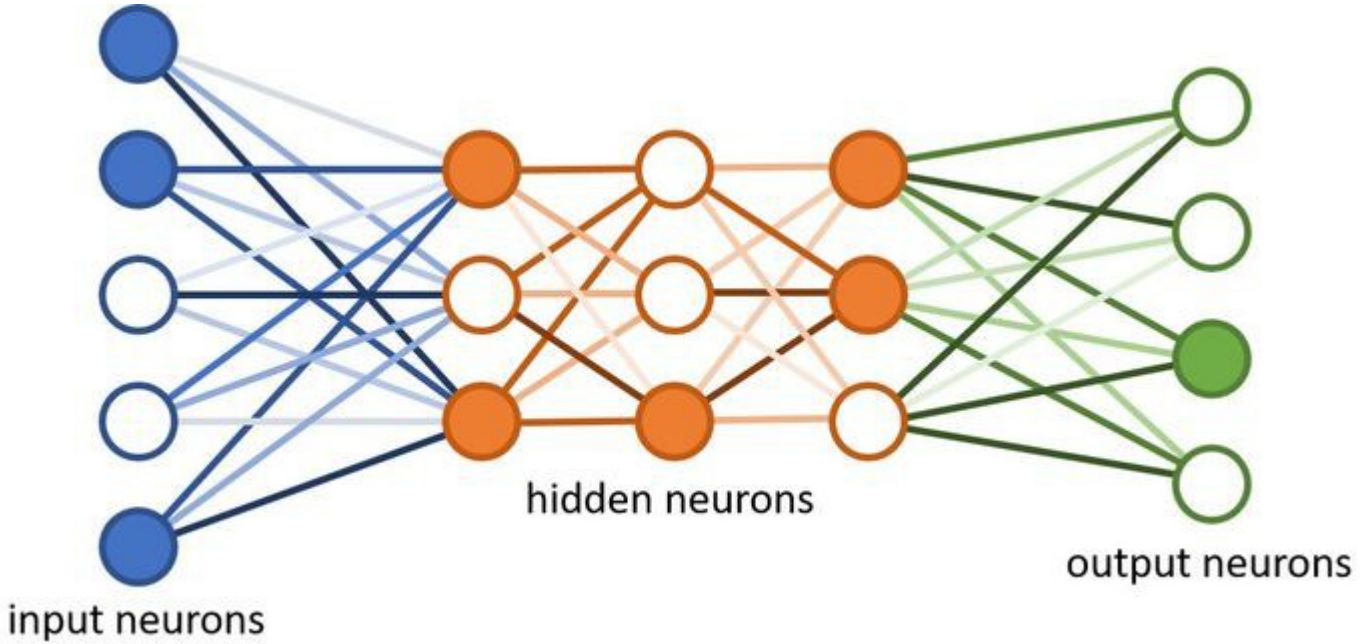
- Réseaux de neurones,
- Machines à vecteurs de support,
- Algorithmes génétiques, ...

Machine learning

Plusieurs catégories d'algorithmes de ML :

- Réseaux de neurones,
- Machines à vecteurs de support,
- Algorithmes génétiques, ...

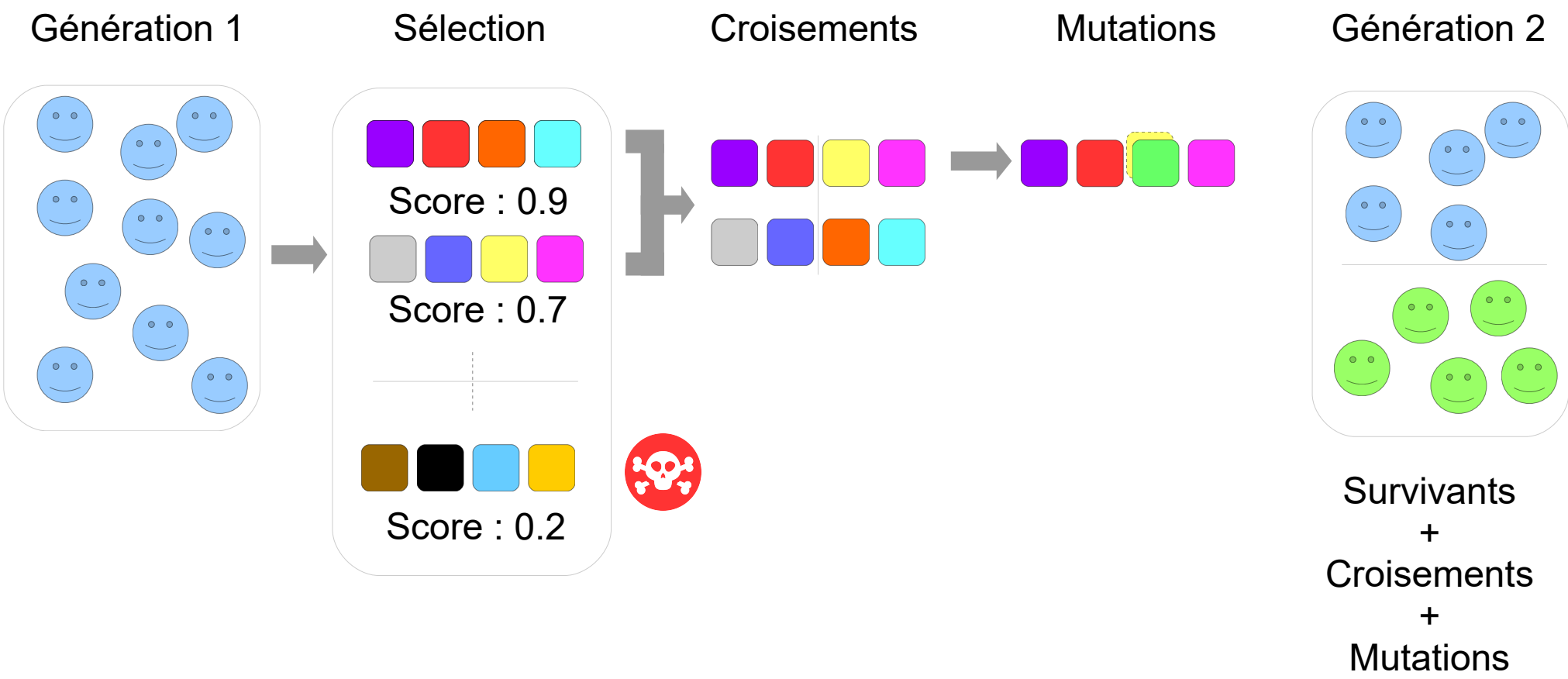
Réseau de neurones



Propagation avant : calcul

Rétropropagation : apprentissage

Algorithme génétique



Machine learning

Deux approches principales d'apprentissage :

- L'apprentissage supervisé.
- L'apprentissage non supervisé

Machine learning

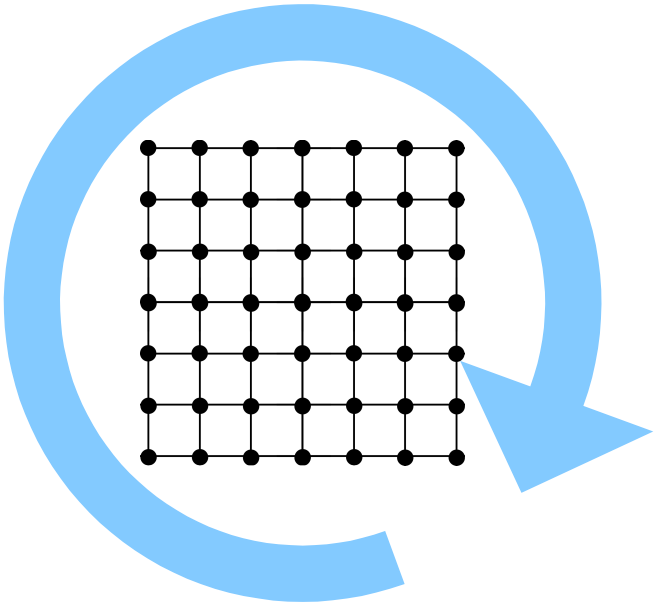
L'apprentissage **supervisé** consiste à fournir des **données étiquetées** à un système pour le calibrer.

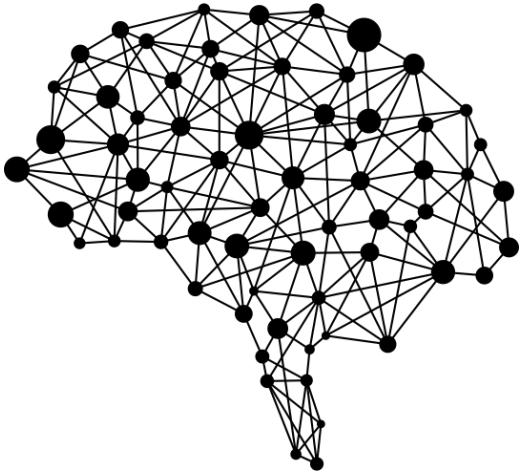
Donnée étiquetée : Donnée pour laquelle la réponse au problème a été renseignée un humain.

L'IA calibre alors un ensemble de paramètres pour que sa réponse corresponde à celle de la "donnée étiquetée".

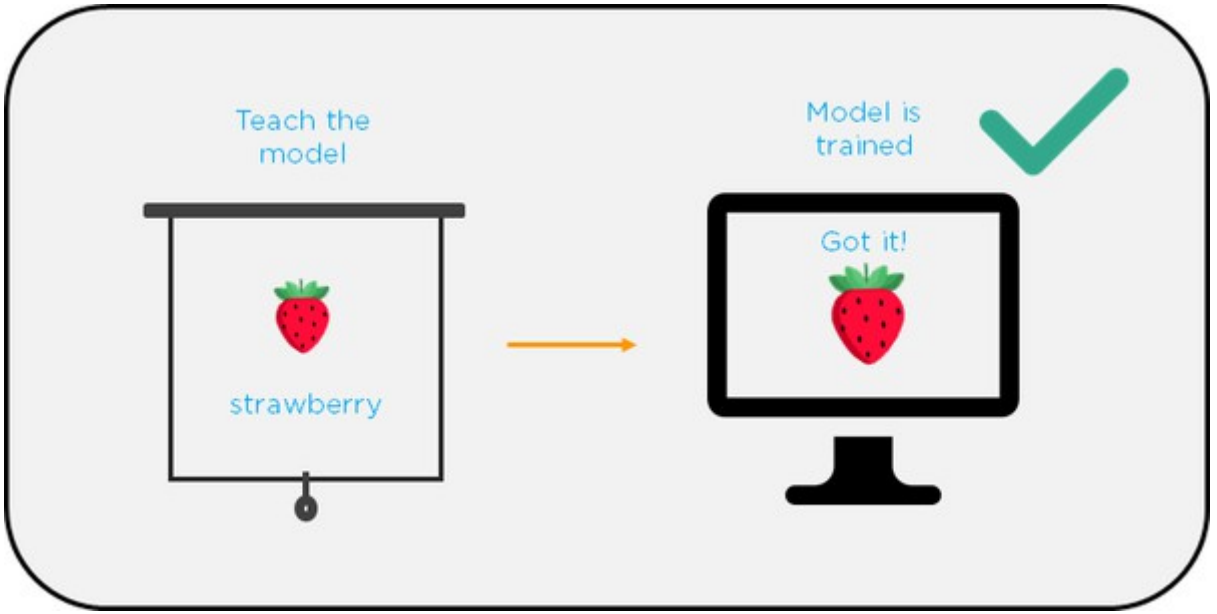
Machine learning

Apprentissage :





Machine learning



Machine learning

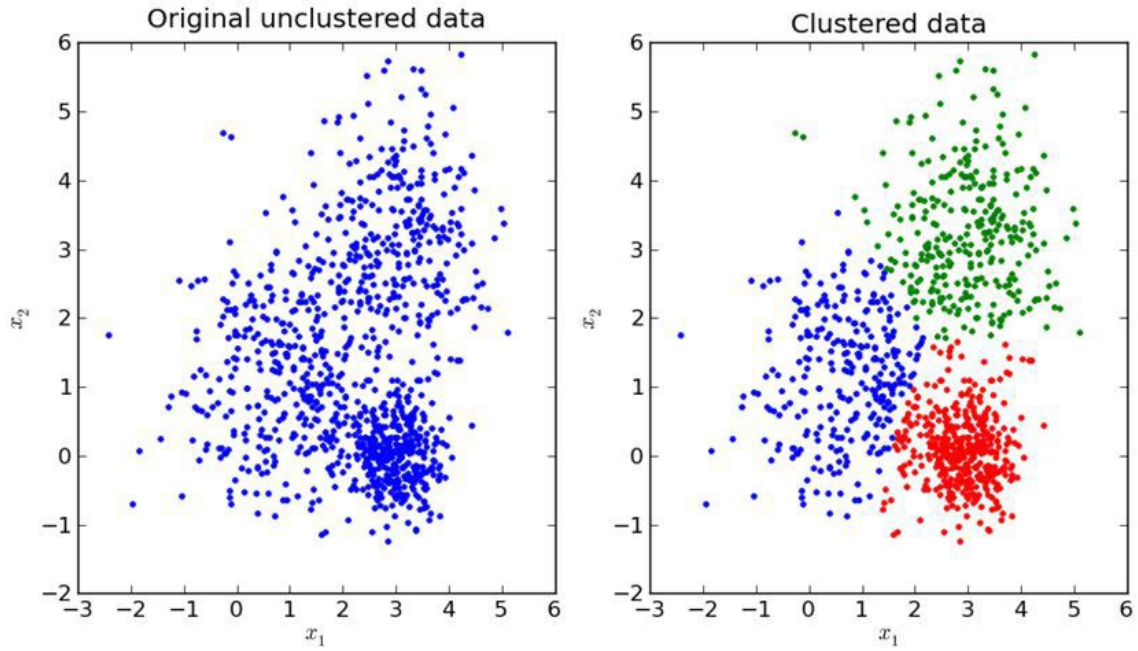
L'apprentissage **non-supervisé** n'utilise pas de données étiquetées.

Il détecte des **recoupements** ou des **anomalies** dans un ensemble de données.

Il permet donc de trouver des solutions aux problèmes que l'on ne se pose pas encore ! :)

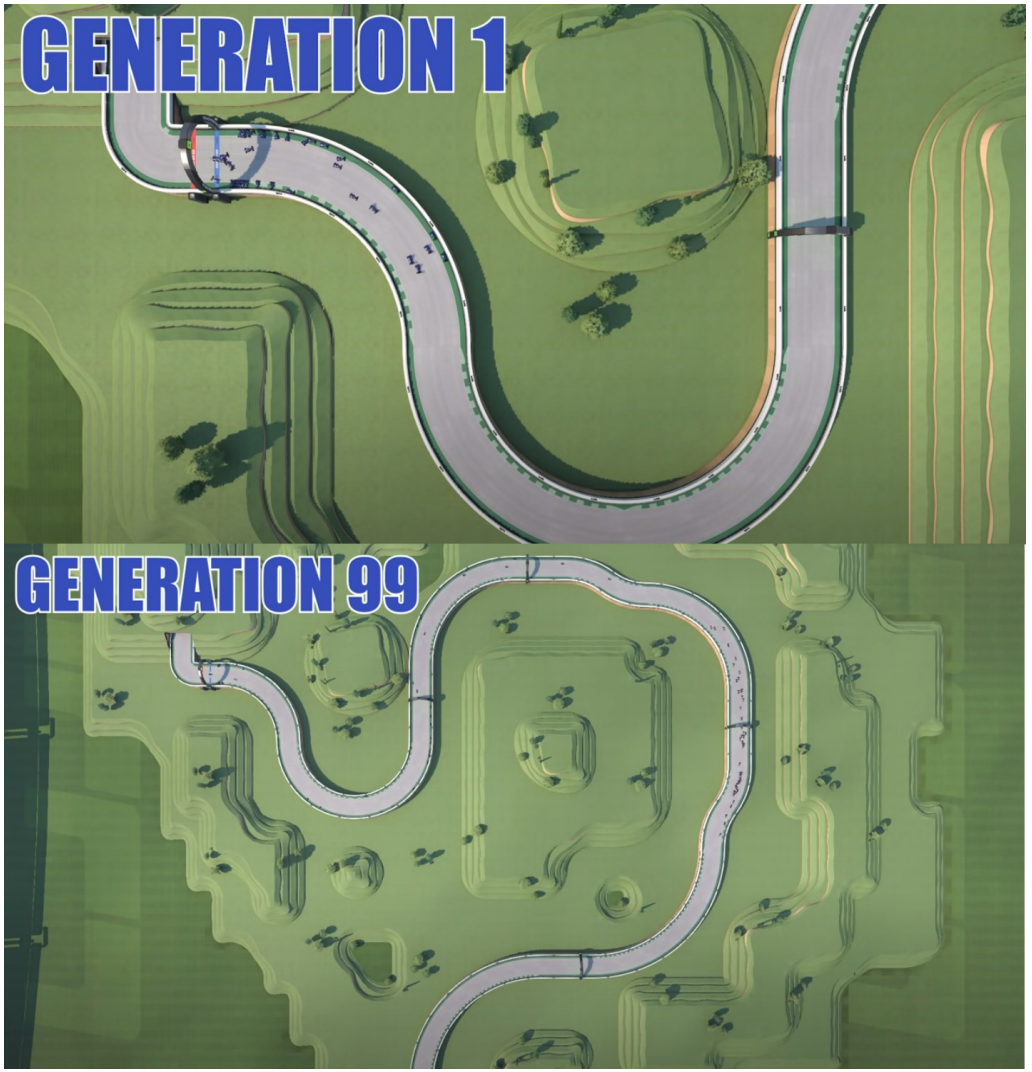
Machine learning

Unsupervised Learning



Trois groupes présentant des similitudes ont été détectés dans ce jeu de données.

Machine learning



Machine learning

Avantage du Machine Learning :

- Indépendant de la complexité,
- Pas de solution explicite.

Inconvénient :

- Répond à un problème unique,
- Difficile à ajuster en fonction du design,
- Demande un grand volume de données.

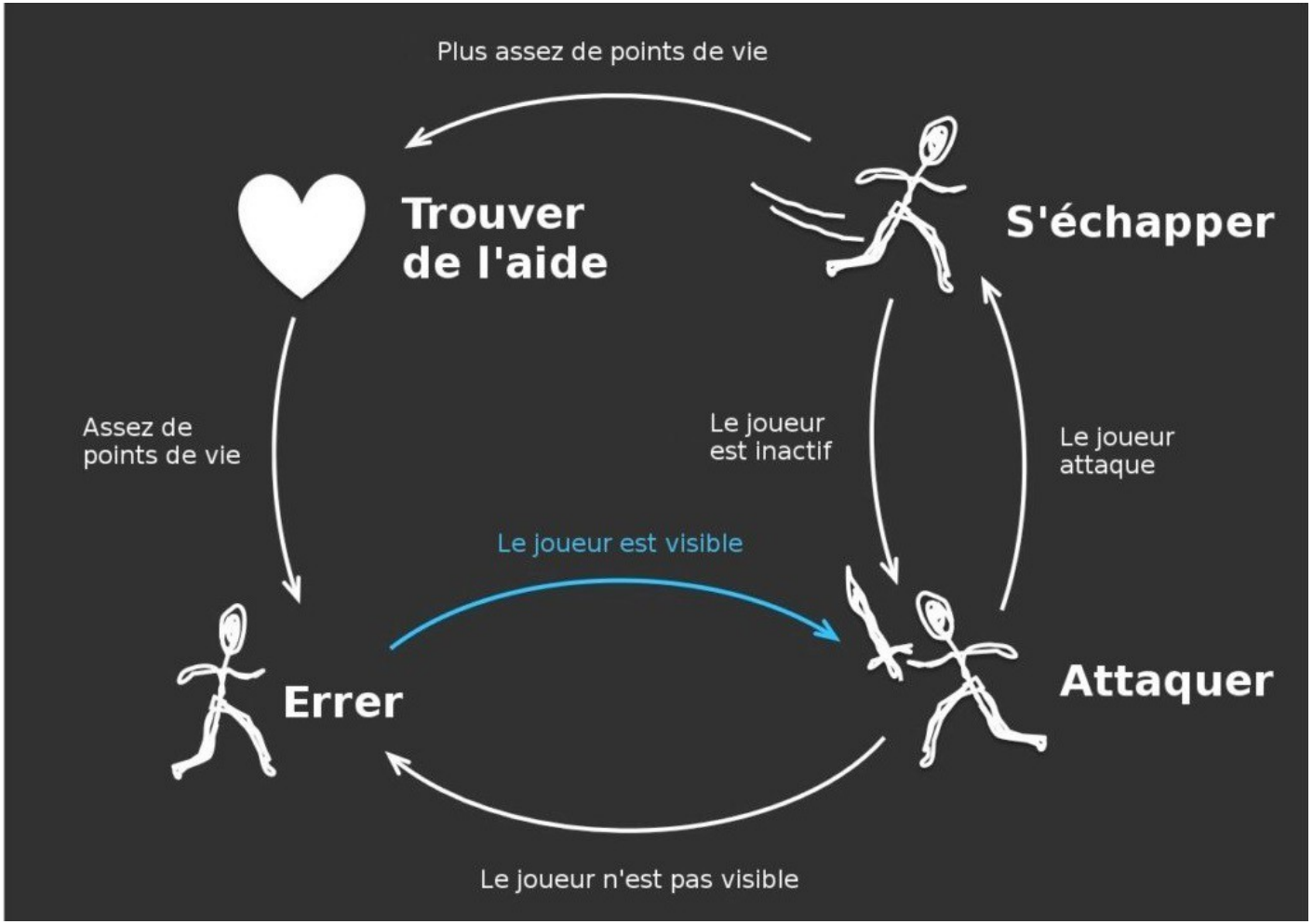
Finite State Machine

State Machine

La **Finite State Machine** (FSM) est l'un des algorithmes les plus simples, clair et performant en termes d'IA "simples" pour le jeu vidéo.

Il est constitué d'un **nombre d'états finis** reliés par des **transitions** conditionnées.

State Machine



State Machine

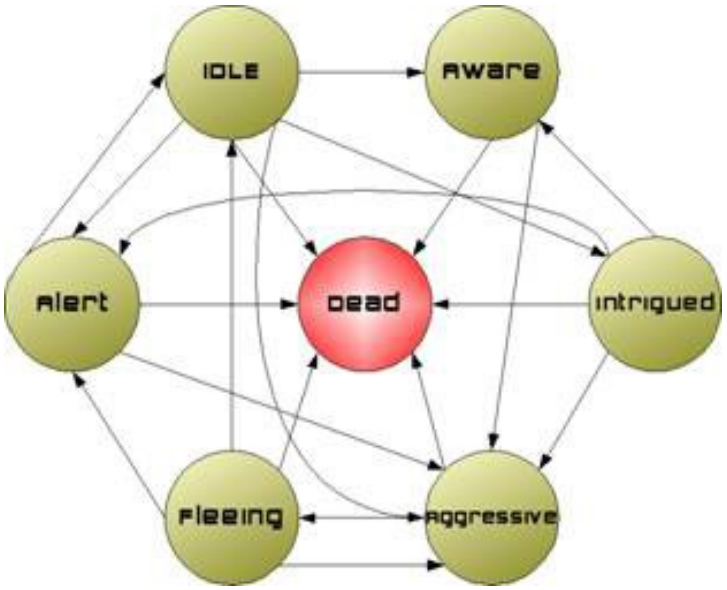
Etat : comportement de l'IA (Attaquer, Fuir, Se soigner, ...),

Transition : Passage d'un état à un autre conditionné par des paramètres,

Paramètre : valeur du contexte du jeu (ennemi à proximité, objets portés, évaluation du terrain, ...)

State Machine

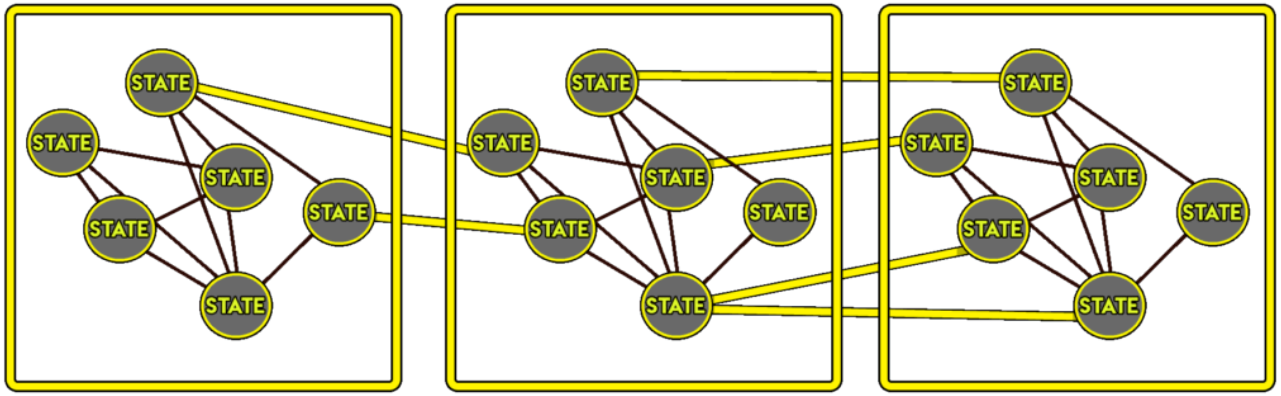
La FSM présente ses limites lorsque le nombre d'états augmente. Les liaisons entre les états se multiplient exponentiellement et la FSM devient peu évolutive.



Beaucoup trop de transitions !

State Machine

Automate fini hiérarchisé (HFSM) : FSM complexe sous-divisée en plusieurs FSM.



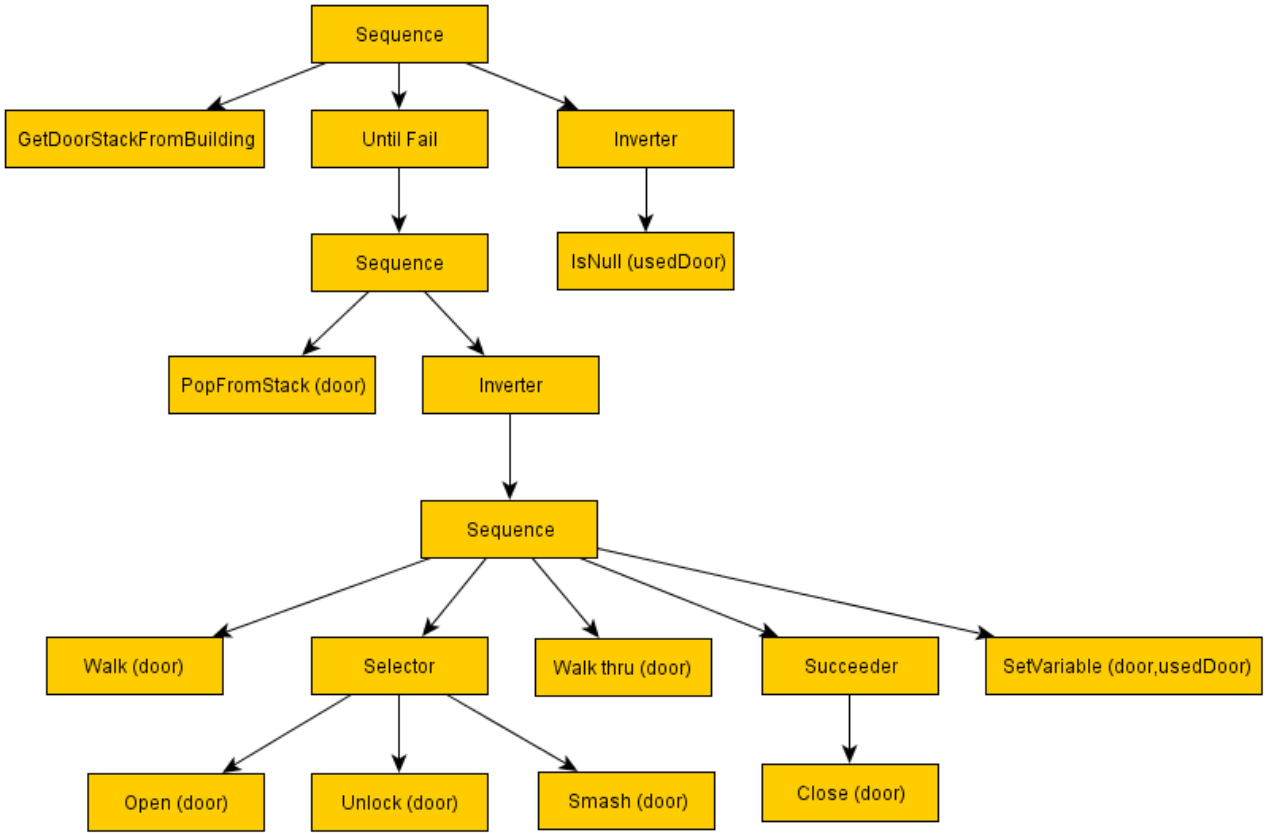
Behavior Tree

Behavior Tree

Le **Behavior Tree** est une approche de l'IA par arbre de choix.

Préféré à la FSM, c'est aujourd'hui l'une des technologies d'IA les plus utilisées dans le domaine du jeu vidéo.

Behavior Tree



Behavior Tree

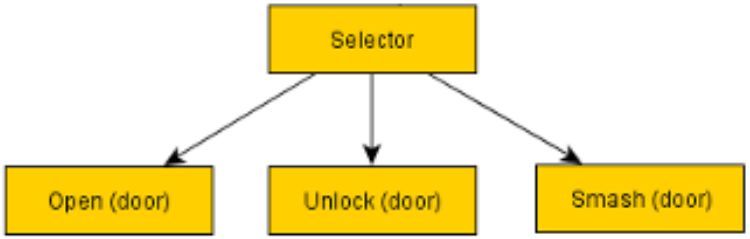
L'arbre est composé de différents types de noeuds :

- Nœuds composés : Un à plusieurs nœuds enfants
- Nœuds décorateurs : Un seul nœud enfant
- Nœuds feuilles : Pas d'enfant

Behavior Tree

Parmi les nœuds composés, on distingue :

- **Le sélecteur** : Il choisit le premier nœud enfant convenant à la situation.
- **Le sélecteur aléatoire** : Il choisit un nœud aléatoire parmi ses enfants.



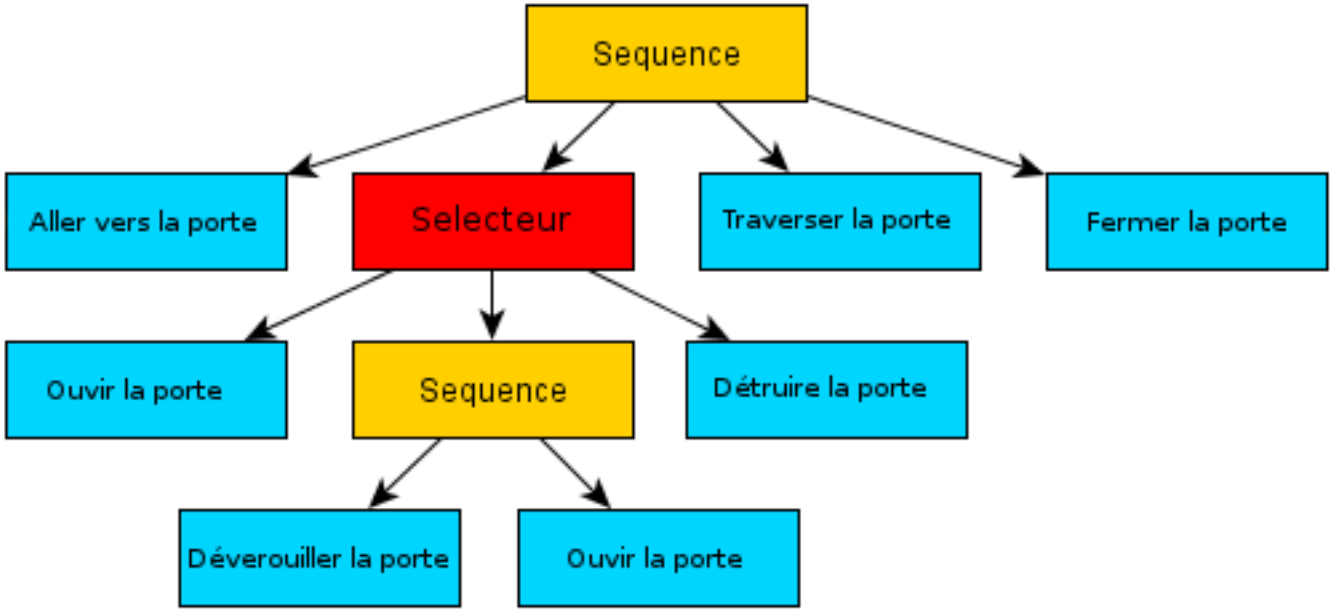
- **La séquence** : Elle exécute successivement chacun de ses enfants. Si l'un des enfants échoue à s'exécuter, elle s'arrête.

Behavior Tree

Les nœuds composés se **lisent de gauche à droite** :

- **Le sélecteur (OR) :** Exécuter le premier enfant possible.
- **La séquence (AND) :** Exécute chaque enfants. S'arrête au premier échec.

Behavior Tree



Behavior Tree

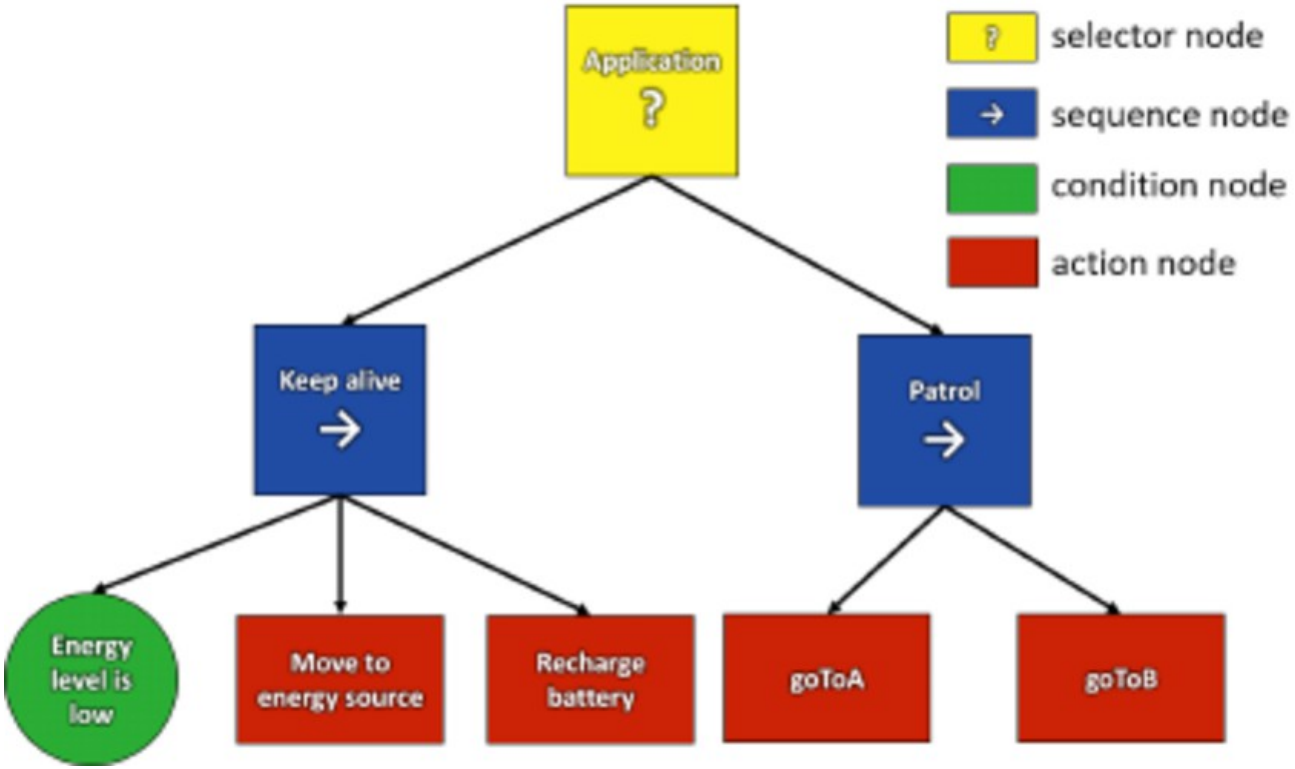
Un **nœud feuille** :

- Effectue une **tâche,**
- Se termine par un **résultat,** succès ou échec.

Types de nœuds feuilles :

- **Action** : S'exécute sur une **durée courte.**
- **Condition** : Conditionne la suite d'une séquence ("Aller au frigo" => **"Frigo plein ?"** => "Manger").

Behavior Tree



Behavior Tree

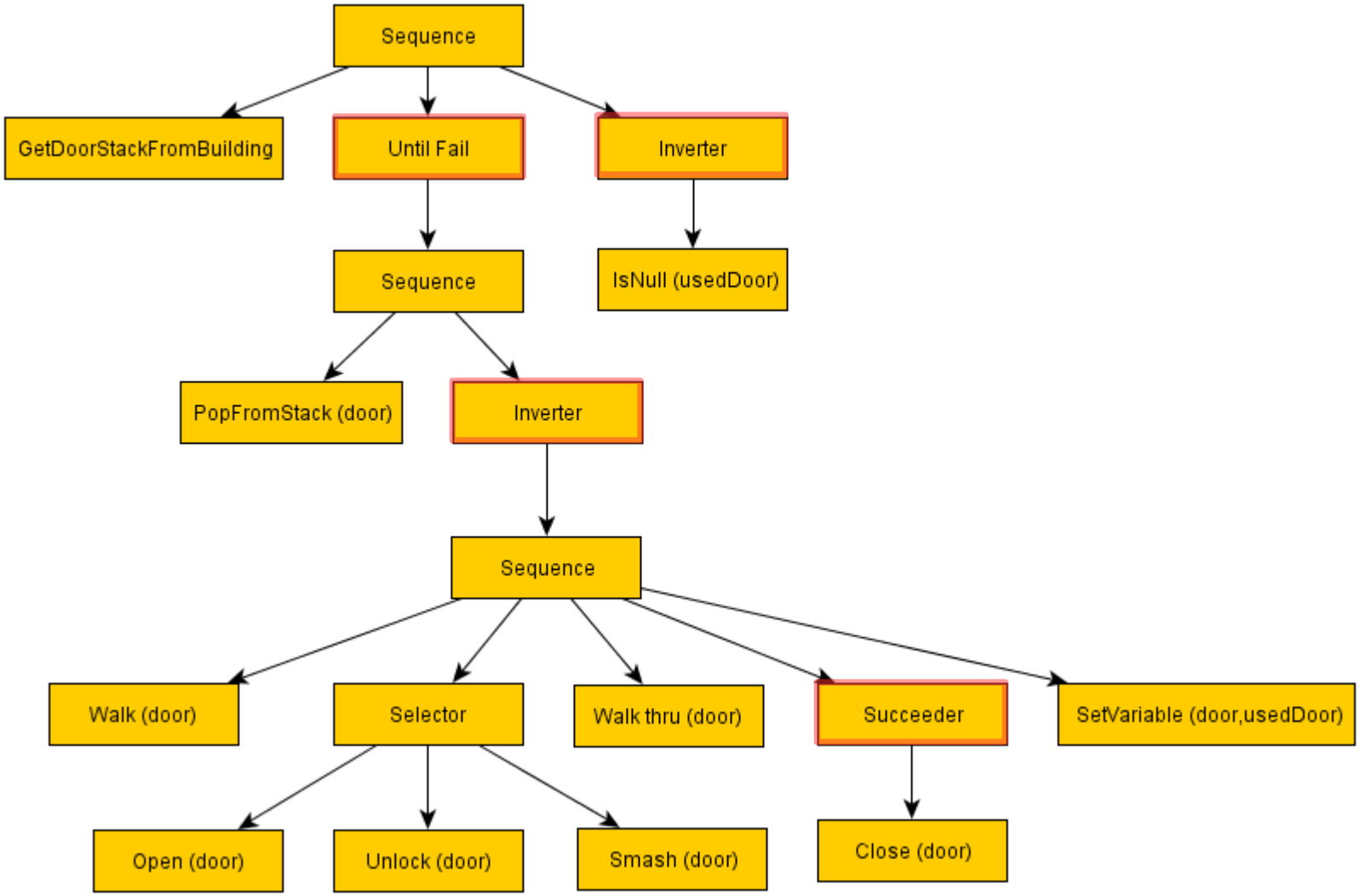
Un nœuds décorateurs peut :

- **Conditionner** l'exécution du nœud enfant
- **Modifier** la valeur renvoiée par le nœud enfant

Exemples de nœuds décorateurs :

- **L'inverteur** : Il inverse le résultat succès/échec de son enfant.
- **Le "succeeder"** : Il change toujours le résultat de son enfant en succès.
- **Le répéteur** : Exécute son enfant un nombre de fois donné.

Behavior Tree



Behavior Tree

Avantages du Behavior Tree :

- Clair
- Modulaire
- Adapté à l'enchaînement d'actions courtes

Inconvénients :

- Peu adapté aux changements fréquents de l'environnement

Utility AI

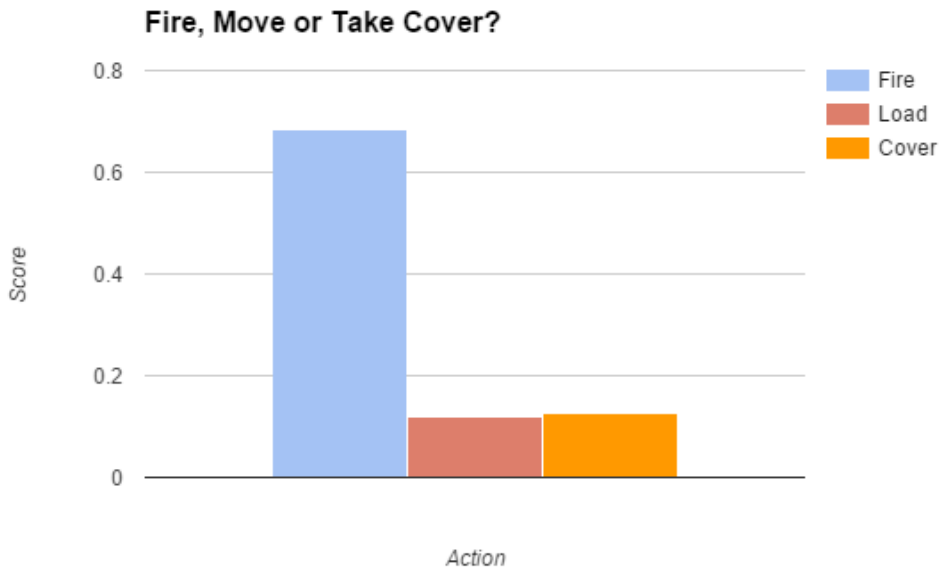
Utility AI

L'IA par utilité, ou Utility AI, est une alternative consistante au Behaviour Tree, offrant plus de réactivité.

Elle consiste simplement à choisir la meilleure action parmi un lot d'actions données.

Utility AI

Chaque action est évaluée périodiquement par une fonction heuristique donnant un score en fonction des paramètres du contexte de jeu.



Utility AI

Une action est composé de scorers. Un **Scorer** :

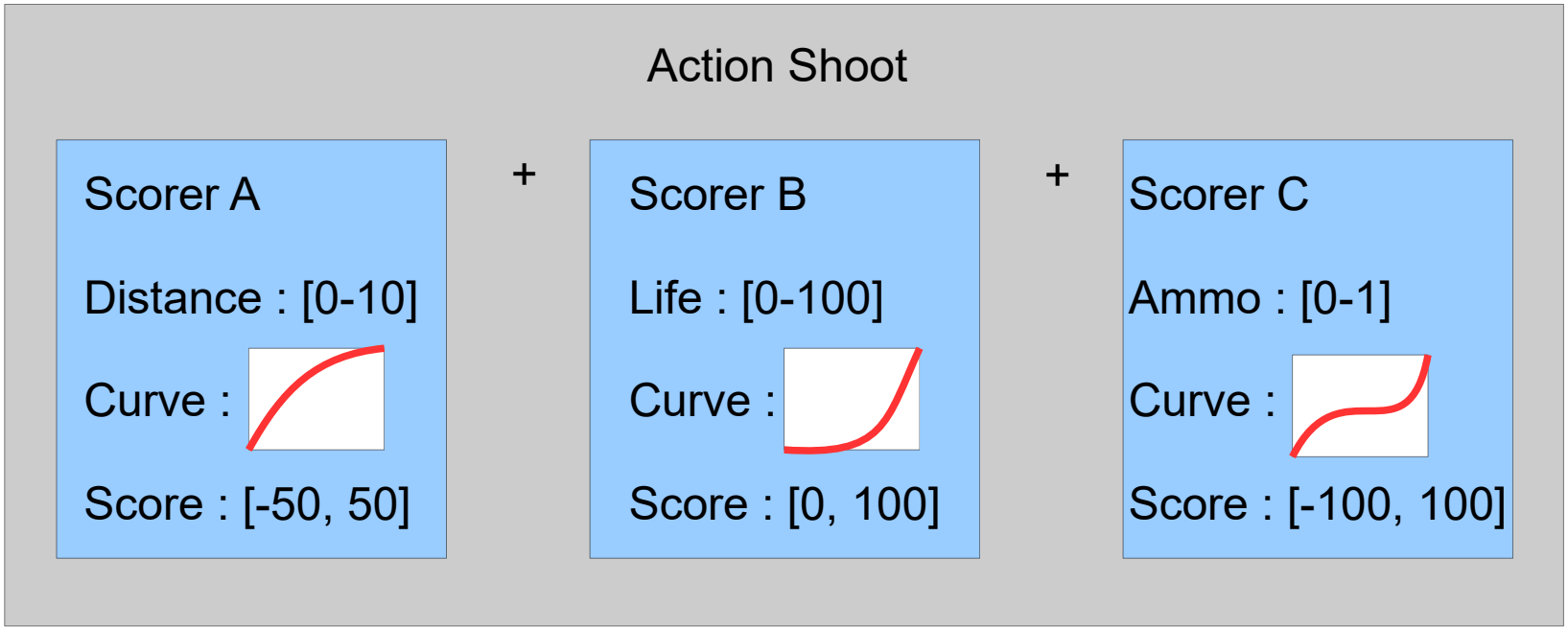
- Prend en entrée une donnée,
- Normalise cette donnée entre 0 et 1,
- Evalue cette donnée sur une courbe entre 0 et 1,
- Interpole ce résultat entre un score min et un score max.

```
nDistance = InverseLerp(distanceMin, distanceMax, distance)
t = scoreCurve(nDistance)
score = Lerp(scoreMin, scoreMax, t)
```

Utility AI

Le score d'une action est le total des scores de ses scorers.

Par exemple, un NPC veux tirer si :



Utility AI

Après évaluation, l'action avec le plus gros score est jouée.

Action	Scorer	Score
Move to Enemy	Distance to Enemy	0-100
	Gun is not loaded	-100
Fire at Enemy	Proximity to Enemy < 50	75
	Cannot make it to cover	50
	Gun is not loaded	-125
Move to Cover	Is not in cover	50
	Proximity to Cover < 50	50
Load	Gun is not loaded	75
	Is in cover	50
	Gun is loaded	-125

Utility AI

Avantage :

- Pas de liens entre les états / actions
- Réactif aux changements d'environnement

Inconvénients :

- Difficile de faire une succession d'actions
- Peu prédictible

Récapitulatifs sur les systèmes de prise de décisions

Récapitulatif

Chaque système est adapté à un besoin (tour par tour, séquence d'actions, réactivité, ...)

Il convient de choisir le système le plus adapté.

Combiner plusieurs systèmes peut permettre de résoudre un problème sur différents niveaux. I.e. : Comportement macro / Action micro.

Termes et Architecture

Agent

Agent : Système d'IA contrôlant un élément de jeu.

L'agent prend des décisions en fonction des informations de son environnement.

Il contrôle un PNJ, un joueur adverse de RTS ou encore le pathfinding d'un personnage.

Blackboard

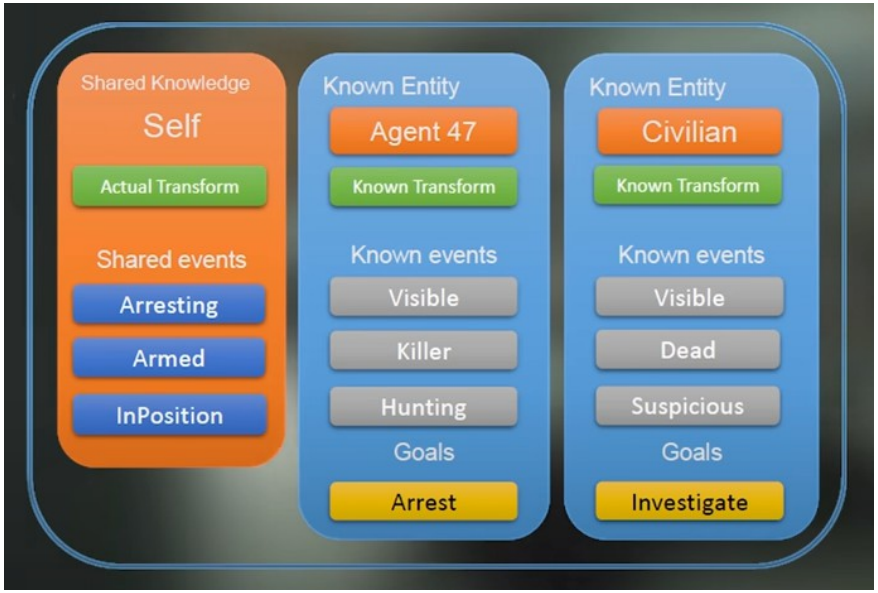
Blackboard : Ensemble d'informations partagées par les états / transitions / actions d'un agent, ou par plusieurs agents.

Le blackboard est mis à jour :

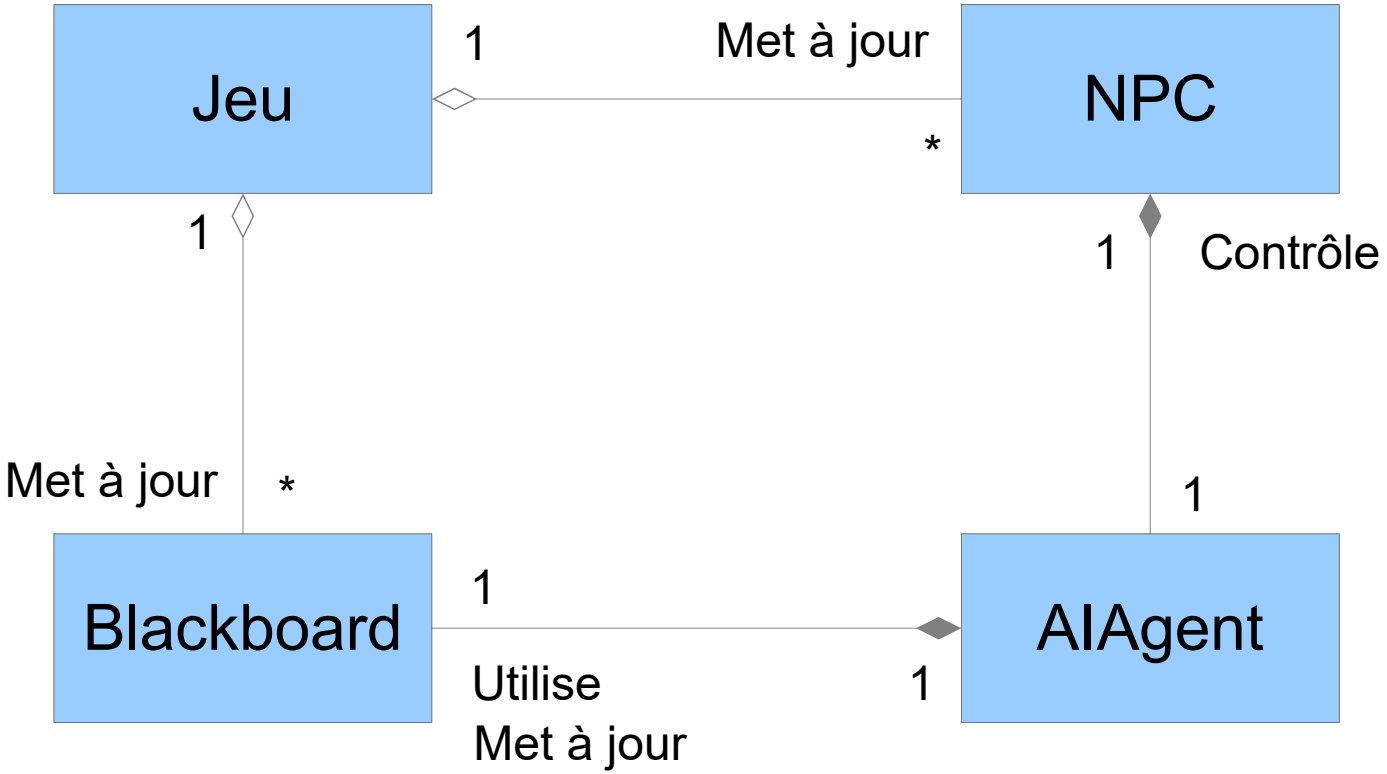
- En interne, par le système d'IA,
- Avec des informations du context de jeu (mise à jour périodique).

Blackboard

Plus les blackboards sont complets et utilisables par les systèmes choisis, plus il sera possible d'affiner l'IA !



Architecture simple



IA Humaine ou Mécanique ?

IA Humaine

L'IA peut être travaillée de manière à réagir de façon humaine :

- Latence
- Aléatoire
- Simulation d'émotions : Colère, surprise, peur, ...

Ce type d'IA sera préféré pour remplacer un joueur dans un jeu multi-joueur.

IA Mécanique

L'IA peut être travaillée de manière mécanique :

- Patterns
- Pas ou peu d'aléatoire
- Mouvements plus rigides

Ce genre d'IA sera adapté aux jeux solo, où l'apprentissage du comportement de l'IA apporte une progression et une satisfaction au joueur.

Quelle IA ?

Le type de comportement de l'IA dépend donc du jeu développé.

L'IA doit avant tout apporter une expérience satisfaisante au joueur.

Elle peut donc être mécanique, voir tricher, pour arriver à l'expérience recherchée.

Références

- The AI of Hitman (vidéo) - AI and Games (et toutes les vidéos de cette chaîne).
- Recreating the False Knight Boss Fight (vidéo) - Dominik Hackl
- Neural Networks (vidéos) – 3Blue1Brown
- Le Deep Learning (vidéo) - ScienceEtonnante