Projet Informatique : Jeu de Coinche

Benjamin DONNOT

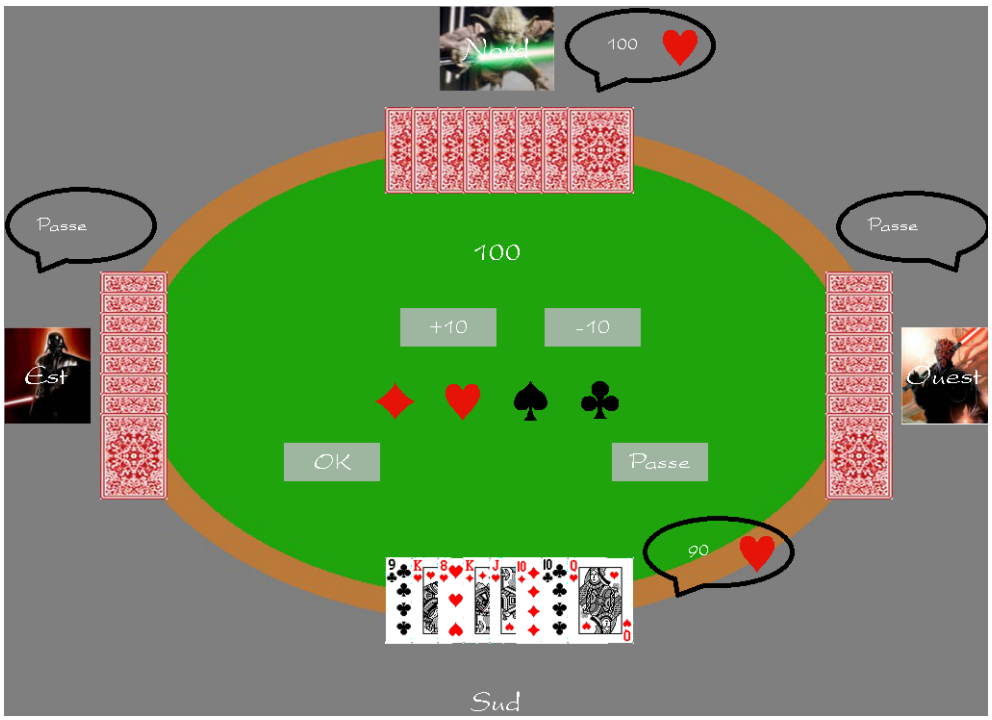

Table des matières

Introduction

Depuis le cours de python de première année, j'ai découvert un champ assez vaste informatiquement : celui de l'*intelligence artificielle*. Ce terme assez générique a plusieurs sens. Ce que j'entends par "intelligence artificielle" dans tout ce rapport, et plus généralement dans le projet tout entier, c'est la capacité pour un ordinateur à prendre des décisions "cohérentes" et ainsi à jouer "convenablement" à la coinche.

La coinche, comme de nombreux jeu de cartes ne se joue pas en information parfaite. Les cartes des adversaires sont cachées ! Outre l'aspect ludique de coder des jeux, ceci peut s'avérer utile "dans la vraie vie". En effet, tous les jours nous sommes amenés à prendre des décisions dans un environnement incertains [1] Résoudre ce type de problème pourrait permettre de faire des grandes avancées dans tout les domaines informatisés, c'est à dire partout...

Ainsi, la recherche informatique se

En 2e année, j'ai pu suivre le cours de C++, et j'ai réalisé un projet de Belote. Je disposais donc d'une interface graphique [2]. Cette partie, indispensable pour les jeux de cartes est très longue à coder, pourtant indispensable pour pouvoir jouer convenablement.
Ne trouvant que peu d'intérêt dans la création d'une seconde interface graphique au cours de ma scolarité, j'ai donc décidé de faire ce projet sur un jeu de Coinche, jeu très proche de la Belote, au moins dans les règles.
Au cours de ma deuxième année, je m'étais concentré sur deux aspects principaux : l'interface graphique, ainsi que la prise. La partie jeu à proprement parlé n'avait été que peu étudiée, et faisait l'objet d'une implémentation rudimentaire. Ce projet aura donc pour but de réaliser une intelligence artificielle capable de jouer de façon "correcte" sur toute une partie de Coinche.

Á de nombreux endroits, dans le code sont présents des commentaires commençant par "TO DO", ceci est dû à de multiples facteurs. J'ai l'intention de continuer à coder ce projet pendant mon temps libre même après ma scolarité de ce projet. J'ai donc laissé consciemment certains aspects à un développement ultérieur. N'étant pas un joueur de Coinche professionnel, une de mes aspirations serait de coder un jeu qui soit au moins aussi fort que moi.

1. Qui est plus nettement plus compliqué que l'environnement simplifié du jeu de Coinche.
2. Celle-ci, même si je l'ai recodée en grande partie n'est pas exempte de défauts, mais ne fait pas l'objet de ce rapport, ni même de ce projet.

I Architecture

II Limitations

pas de tout atout sans atout

Belotte non prise en compte

prise rudimentaire

encore beaucoup de "TO DO" dans le code, parce que ce projet n'a pas vocation a être arrêté pour la validation de ce cours.

III Interface graphique

multi-threading :-)

upgrade to SDL 2.0, voire SFML :-)

IV *Intelligence Artificielle* "basique"

1 Aléatoirement

On joue une carte aléatoire parmi celles qu'on a le droit de jouer :-)

benchmark : référence de jeu.

2 Avec des scores

score de chaque carte jouable en fonction de conditions, on joue la carte avec le plus gros score.

prise en compte des appels, via la classe "Memory" (template)

V *Intelligence Artificielle* par évaluation Monte Carlo

1 Donner les cartes

problème : respect de l'aléa final : il faut une certaine uniformité dans quand on donne les cartes (cf. pistes d'améliorations)

autre difficulté : respecter les contraintes stockées dans la mémoire.

2 Jouer les jeux

3 Piste d'améliorations

Importance Sampling : regarder en détail les actions de chaque pour 'sampler' de façon plus convenable. Ceci permettrait de tirer parti d'informations telles que 'si un joueur a joué ça, c'est qu'il probablement avoir ça' [cas des appels]. Alors que la on ne tire parti que des infos du genre "un joueur a joué ça, il ne peut pas avoir ça".

Prime à la découverte / communication avec le partenaire : pour l'instant l'IA ce sont deux autistes qui prennent des décisions. On pourrait imaginer mettre en place une réelle communication par les cartes (appels).

optimisation / profiling : trouver un outil pour rendre l'évaluation plus rapide ce qui permettrait de faire plus d'évaluations, donc d'être plus performant :-)

4 Performance

Méthodologie : l'IA qui s'affronte elle même.
Deux équipes IA de types différents.
cas 1 : random vs score cas 2 : Monte Carlo random vs score cas 3 : Monte Carlo score vs Monte Carlo random

Conclusion