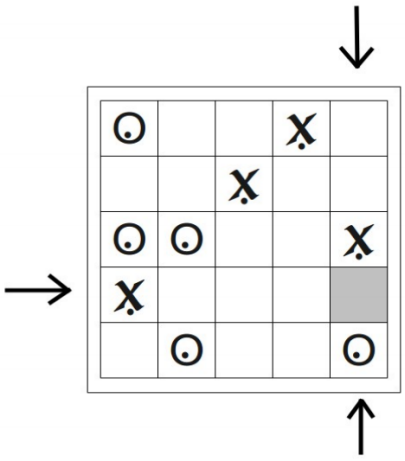
Application de conception
Développer un jeu quixo

Résumé

L'objectif de l'application est de réaliser un jeu quixo en utilisant un langage de programmation objet doté d'une intelligence artificielle (algorithme : *min-max* ou *alpha-beta*). Pour ce faire, les méthodes liées à la programmation orientée objet et modélisation du logiciel devront être appliquées. Vous devez donc prendre en compte toutes les subtilités liées au jeu quixo (condition de déplacement des pions, de gain de la partie...) ainsi que réaliser les différentes fonctionnalités définies dans la suite de ce document.

Description succincte du jeu quixo

Le but du jeu est de réaliser une ligne de 5 ronds ou de 5 croix, à l'horizontal, la verticale, ou en diagonale. En début de partie, les 25 cases du jeu sont remplies avec des pions « blancs ». Chaque joueur choisi un symbole : rond ou croix. A tour de rôle, ils prennent l'un des 16 pions sur le contour du plateau, de leur symbole choisi au départ ou un pion « blanc ». On ne peut pas prendre un pion ayant le symbole de l'adversaire. Si le pion est « blanc » on le transforme ayant son propre symbole. Le pion saisi doit être replacé sur le plateau « en poussant », à l'une des extrémités d'une rangée incomplète crée lors du choix du pion (voir figure ci-dessous).



Le pion ne peut pas être reposé sur la case où il a été pris. La partie se termine lorsque l'un des joueurs réalise une ligne de 5 pions.

Il existe sur Internet plusieurs vidéos expliquant les règles du jeu :

- https://www.youtube.com/watch?v=tU5wycG8SHU
- https://www.youtube.com/watch?v=cZT5N6hIFYM
- https://www.youtube.com/watch?v=KHrAbSMAkmo

Source : Règle du jeu Quixo

Conception Orientée Objet

L'objectif de ce travail est de concevoir une modélisation de ce jeu afin de pouvoir ensuite l'implémenter. Vous devez donc réaliser un diagramme de classe UML intégrant les contraintes du jeu ainsi que les fonctionnalités devant apparaître dans le logiciel. Attention, cette étape de modélisation est cruciale. Ne négligez donc pas cette étape, quitte à réaliser d'autres types de diagrammes pour comprendre le problème (diagrammes de cas d'utilisation, d'états-transitions…). Vous devez mettre en place 3 différents designs patterns vu en modélisation du logiciel (à noter que le design pattern Modèle-Vue-Contrôleur est à mettre en place obligatoirement). A vous de voir/juger quel design pattern est le plus adapté au jeu d'échec.

Fonctionnalités attendues

Le jeu devra être développé dans un langage de programmation orienté objet et sera, bien entendu, basé sur la modélisation UML que vous aurez proposée (voir partie précédente). Votre code devra être, lorsque cela est nécessaire, commenté, notamment dans l'optique d'être exporté avec un outil de génération de documentation automatique (par exemple doxygen). L'étape de modélisation est également un travail personnel, vous vous apercevrez que plusieurs modélisations sont possibles : il est donc fort peu probable que vous ayez la même que votre voisin.

Les différentes fonctionnalités attendues sont :

- Concevoir le moteur du jeu, c'est-à-dire l'échiquier, les pions (et leurs déplacements).
- Vérifier que vous avez bien mis en place 3 différents designs patterns vu en modélisation du logiciel.
- Vérifier que le jeu se termine bien (partie gagnée ou perdu).
- Un affichage du jeu doit être prévu, et doit permettre l'interaction. Cela peut se traduire par un mode console (a minima) ou par une interface graphique (**très fortement recommandé**).
- Une intelligence artificielle basé sur un algorithme min-max ou alpha-beta (voir l'annexe). A noter que l'algorithme min-max ou alpha-beta devra être multi-threadé.

Rendu du projet

À la fin de ce projet, il est attendu :

- Un rapport intégrant votre analyse du problème, avec notamment votre diagramme de classe commenté précisant vos choix de conception. Vous décrirez également les différentes fonctionnalités logicielles que vous avez développées, les difficultés rencontrées, ainsi que les améliorations possibles de votre programme.
- Le code source de votre travail, bien entendu commenté. L'enseignant responsable du TP devra également avoir vu votre programme fonctionner.
- La présentation orale de votre travail, avec 5 minutes de présentation et 5 minutes de questions.

Attention : de nombreuses ressources et programmes (de qualité variable) sont disponibles très facilement, notamment sur Internet. Vous pouvez bien évidemment les consulter, mais gardez bien à l'idée que c'est un travail personnel : l'enseignant responsable sera particulièrement vigilant sur ce point et prendra les sanctions adéquates en cas de détection.

Annexe : Stratégie du Min-Max dans un jeu

Jeux étudiés

Nous nous intéresserons aux jeux à deux joueurs, jouant à tour de rôle, sans intervention du hasard. Exemples : Go, Dames, Échecs, Othello, etc. jeux qui sont appelés jeux de Nim.

Problème :

Partant d'un état donné du jeu, comment chaque joueur peut-il choisir son coup, de façon à gagner à coup sûr, ou au moins, à augmenter considérablement ses chances de gains.

Un jeu de Nim, peut toujours être représenté par un arbre. Dans cet arbre, un nœud E représente un état du jeu, ses fils sont les divers états susceptibles d'être atteints en un seul coup à partir de E.

Dans l'arbre complet du jeu, chaque feuille sera un état final, succès ou échec pour le joueur considéré. Si le joueur connaît cet arbre complet, il peut toujours se placer sur le chemin du succès.

Malheureusement, un tel arbre est très vite gigantesque, et sauf cas très simple, il ne sera pas construit dans sa totalité. Nous nous contenterons d'un arbre partiel, qui nous permettra de déterminer, non un coup gagnant, mais seulement un meilleur coup, augmentant nos chances de gain.

Présentation du Min-Max

Construction de l'arbre :

Partant d'un état quelconque du jeu, nous supposons construite une partie de l'arbre décrivant les futurs états possibles. Nous dirons qu'un arbre a une profondeur P, s'il contient P niveaux en plus de la racine. Cette profondeur sera en général variable, et conditionné par :

- la place mémoire disponible,
- le temps de réponse désiré,
- la compétence recherchée,

toutes fonctions qui croissent en même temps que P.

Nous appellerons PLUS et MOINS les deux joueurs en compétition. Nous prenons pour exemple le jeu de Tic-Tac-Toe dons nous présentons un arbre de profondeur 2, construit à partir de l'état initial de ce jeu.

L'idée est d'associer à chaque état suivant possible, une valeur numérique V telle que en comparant deux valeurs Vi et Vj , on sache si l'état Ei est préférable à l'état Ej ou non.

Pour calculer cette valeur V on utilisera :

- une fonction d'évaluation permettant d'associer une valeur aux feuilles de l'arbre étudié,
- une technique permettant de calculer la valeur d'un état E, quand sont connues les valeurs de tous ses fils E1, E2, …, En.

La fonction d'évaluation :
Elle vérifiera toujours les conditions suivantes :

Les états feuilles du jeu qui semblent plus favorables à MOINS recevront une valeur négative, d'autant plus petite que l'état paraît plus favorable à MOINS.

Les états feuilles du jeu qui semblent plus favorables à PLUS recevront une valeur positive, d'autant plus grande que l'état paraît plus favorable à PLUS.

En particulier, si une feuille est un état final du jeu, sa valeur sera :

- $+\infty$ si l'état est gagnant pour PLUS,
- $-\infty$ si l'état est gagnant pour MOINS.

Toutes les autres valeurs apparaissant dans l'arbre sont calculées à partir des valeurs données aux feuilles. La réussite de notre méthode dépend donc directement de la définition choisie pour cette fonction.

Dans l'exemple traité (Tic-Tac-Toe), nous pourrions prendre la définition suivante pour la fonction d'évaluation :

F(E) = nombre de lignes, colonnes et diagonales ouvertes pour PLUS – nombre de lignes, colonnes et diagonales ouvertes pour MOINS.

Remontée des valeurs vers la racine
Plaçons-nous dans le cas ou PLUS cherche quel "meilleur coup" il doit jouer.

Soit E l'état à partir duquel il doit jouer. Pour trouver le meilleur coup il doit regarder tous les fils de E, et retenir celui qui lui est le plus favorable.

On dira que ce nœud est un nœud MAX car PLUS prendra en compte les valeurs trouvées pour chacun des fils de E, et conservera la valeur MAXimale pour l'affecter à E.

Par contre, pour évaluer un des fils de E, soit E', PLUS doit prendre en compte tous les coups que MOINS pourrait jouer, et retenir plus particulièrement le coup le plus favorable à MOINS (qui est le plus défavorable à PLUS).

On dira que un tel nœud est un nœud MIN car PLUS s'intéresse à tous les fils de E', et retient celui qui a la valeur MINimale pour l'affecter à E'.

D'après ces définitions on voit qu'un arbre tracé par le joueur PLUS, est formé alternativement de nœud MAX puis MIN, qu'il évaluera en prenant successivement un maximum puis un minimum, d'où le nom donné à cette stratégie.

Exemples :

Nous utiliserons le jeu de Tic-Tac-Toe pour illustrer notre méthode. Dans ce jeu, chaque joueur pose un pion dans une case libre du tableau, le gagnant étant le premier joueur qui aligne trois pions.

Programmation du MIN-MAX

Idées générales

En matière de programmation, nous retiendrons de l'étude précédente que la recherche d'un meilleur coup se fait en effectuant un parcours en post-ordre dans l'arbre des états possibles. L'arbre considéré ne sera jamais codé comme tel en mémoire. La procédure récursive définissant le parcours en post-ordre sera chargé de construire à chaque pas l'état associé au nœud étudié, une fois ce nœud évalué, il sera oublié.

Pour descendre dans l'arbre, il faut simuler le jeu des coups successifs à partir de E. Décrire l'état du jeu à l'aide d'une variable locale serait la solution idéale, mais, si cet état est décrit dans une matrice, un problème d'encombrement se présente très vite. Nous utiliserons donc une seule structure globale pour décrire les divers états rencontrés, elle sera successivement modifiée puis rétablie dans son état initial.

La procédure de parcours a pour tâche de calculer la valeur d'un nœud (le nœud courant). Cette valeur sera soit fournie par la fonction d'évaluation, soit obtenue à partir des valeurs des nœuds fils (calculées par appels récursifs).

Remarque :

Aucun coup exceptionnel n'a été pris en compte. Les deux plus fréquents (selon les jeux bien sûr) étant :

- bien que l'état du jeu ne soit pas final, l'un des deux joueurs ne trouve aucun coup possible et doit passer son tour,
- inversement, un joueur ayant choisi de jouer un coup C, ce coup (ou l'état du jeu) l'autorise à jouer immédiatement un nouveau coup, avant de passer la main à son adversaire.