— Algorithmique & Programmation – D.E.S.S. Imoi —

Projet : des adversaires intelligents

Comme pour le projet de l'an dernier, le but de ce projet est de vous faire découvrir comment un ordinateur peut être programmé pour jouer intelligemment à un jeu. Ce projet est cependant moins ambitieux que celui de l'an dernier, et ce à deux niveau :

- le jeu considéré a déjà été étudié et programmé au cours du dernier TP, et
- ce jeu est beaucoup plus simple que ceux considérés dans le projet de l'an dernier, ce qui simplifie d'autant la stratégie à définir pour le joueur ordinateur.

Malgré tout, on souhaite utiliser autant que possible les possibilités offertes par l'héritage en C^{++}, de façon à minimiser le travail à effectuer pour ajouter un joueur ordinateur intelligent au jeu déjà défini. Ce projet sera découpé en trois étapes :

1. finir la programmation du jeu avec des joueurs ordinateur simples (travail du TP 4),
2. définir les classes (et leurs méthodes) nécessaires à la mise en place des joueurs ordinateur intelligents, et enfin
3. programmer ces classes et vérifier leur fonctionnement.

1 Fin du jeu simple

La première version du jeu doit être terminée aussi vite que possible, et de toute façon avant la fin de cette semaine. Les fichiers source (d'extension `.cc`) qui ont été créés (et les autres fichiers modifiés, s'il y en a) seront envoyés en documents attachés à l'adresse `Alexis.Scheuer@loria.fr`. N'oubliez pas de préciser le sujet de votre courrier électronique : par exemple, `AP, TP4 ...` où les ... sont remplacés par le(s) nom(s) de l'auteur (des auteurs).

2 Partie algorithmique

L'étape suivante consiste à réfléchir à la façon dont on va pouvoir améliorer le comportement des joueurs ordinateur. En effet, à la fin de la partie précédente, ceux-ci choisissent aléatoirement la carte de leur main qu'ils vont chercher à marier et le joueur auquel il vont faire leur demande. Pour agir plus intelligemment, c'est-à-dire comme un humain le fait dans ce jeu, il faut mémoriser les demandes qui ont été faites : elles indiquent que le demandeur possède le complément de la carte demandée. Cette information peut ensuite être utilisée, lorsqu'on possède la carte demandée, pour faire des mariages.

Votre travail consiste dans un premier temps à réfléchir à la mise en œuvre de cette amélioration des joueurs ordinateur. Pour cela, vous allez devoir modifier certaines classes définies pour le jeu, et en ajouter d'autres. Pour ces modifications et ajouts soient les plus judicieux possible, il est préférable de commencer par y réfléchir formellement par écrit.

.../...

On vous demande donc, dans cette première étape, de préciser les méthodes que vous souhaitez ajouter aux classes existantes, et de décrire les nouvelles classes : représentation et méthodes, mais surtout relation et nature (publique, protégé ou privé) des éventuels héritages. La hiérarchie des classes sera illustrée par une figure donnant le graphe d'héritage et celui des dépendances. On donnera ensuite, pour chaque nouvelle méthode de chaque classe, l'en-tête de cette méthode soit en C^{++} (par exemple, `char toto(const int) const`), soit clairement (la fonction `toto` prend comme paramètre un argument entier non modifié, et retourne un caractère sans modifier l'objet sur laquelle elle est appelée). Cet entête devra être suivi d'un algorithme précisant le fonctionnement de la méthode. Il n'est pas nécessaire de trop détailler cet algorithme, mais il faut y avoir suffisamment réfléchi pour ne pas découvrir ensuite (dans la partie programmation) qu'il ne fonctionne pas.

Cette étape donnera lieu à la rédaction d'un rapport, dactylographié ou **rédigé très proprement** (attention, la présentation entrera en compte pour la note). Ce rapport devra être rendu avant les vacances de Noël. En cas de question quant à la constitution de ce rapport, ne pas hésiter à me contacter.

3 Partie programmation

La dernière étape de ce projet consistera à programmer et à tester les méthodes et les classes mises au point dans la partie algorithmique. Cette partie donnera lieu à l'élaboration d'un dossier regroupant les fichiers en-tête des classes, leur fichiers source, ainsi que les fichiers de test et leur utilisation : les premiers définissant les fonctions de test, il faut aussi décrire comment utiliser ces fonctions, c'est-à-dire donner de **nombreux** exemples de tests et leur résultats. Un énoncé précisant votre travail sera distribué avec le corrigé de la première partie.

Attention : la notation de cette partie sera fortement influencée par les deux critères suivants :

- la lisibilité des fichiers en-tête et source, et en particulier la présence et la précision des commentaires,
- la qualité et la variété des tests effectués pour vérifier le bon fonctionnement des classes définies.