Projet informatique de L2S4

Responsable : Vincent Dugat

15 février 2017

Le projet consiste en un travail de programmation à effectuer. En plus de ce travail, le projet intègre une étude préliminaire de type génie logiciel et les documents correspondants, et un cadre de gestion et de suivi de projet. Le sujet est présenté en annexe et dans un document technique séparé.

Attention ! Des parties du projet (voir section 4) concernent directement les UE (SD, POO et BD). Elles seront notées (voire rendues) séparément par les les enseignants de ces UE.

1 Organisation générale

Le travail sera réalisé en équipes de 2 ou 3 personnes. Le faire seul est risqué (beaucoup de travail pour une personne), de même que le faire à quatre (coordination plus compliquée).

Un encadrant sera affecté à chaque équipe. Il représentera le client et aussi partiellement le chef de projet.

Le projet est constitué d'un programme informatique, de documents techniques et de fiches d'avancement de projet à rendre à intervalles réguliers.

Il sera programmé en langage C et en Java. Vous trouverez des informations pour interfacer les deux langages dans l'annexe du document technique qui accompagne le sujet.

Le calendrier du projet est sur Moodle

2 Travail à effectuer dans le cadre de l'UE projet

2.1 Gestion de projet

Vous devez fournir au client (votre tuteur) certains documents classiques dans le développement de projet en utilisant la notation UML. Certains de ces documents concernent l'équipe, d'autres sont individuels (10 points sont attribués à cette partie) :

- Cahier de charges (Equipe) - 1,5 points : Ce que vous avez compris du sujet et qui va être programmé. Ce document doit être validé par le client.
- Dossier d'analyse et planning (Equipe) - 1,5 points : Description des modules/paquetages principaux et planning des principales échéances des principaux livrables.
- Déroulement du projet (Individuel) - 3 points pour l'ensemble des fiches : 3 fiches d'avancement de projet décrivant l'organisation de l'équipe ainsi que la répartition et le déroulement des différentes tâches.
- Bilan (Equipe) - 1 point : Document décrivant si vous avez tenu vos objectifs accompagné d'un résumé des problèmes rencontrés et solutions développées, autant pour la partie logicielle que pour la partie gestion du projet.

- Recette/Validation (Equipe/Individuel) - 1,5 point : présentation du programme au client, chacun présentant plus en détail la partie qu'il a développée. Echange de questions et réponses avec le client.
- Présentation orale (Equipe/Individuel) - 1,5 points : Présentation de l'équipe synthétisant le déroulement du projet (prévisions, résultat, bilan) devant un public constitué de votre tuteur et de vos pairs.

Les critères de notation pour les documents sont :
- qualité des écrits (lisible, compréhensible, informatif)
- maîtrise du langage UML pour présenter les parties techniques
- respect des échéances

2.2 Programmation (10 points)

Le programme que vous allez rendre sera classé dans l'une des catégories détaillées ci-dessus et sera noté par rapport à la note maximale associée. Dans les 3 versions vous devez utiliser divers modules en Java et en langage C (.h et .c). Vous devez utiliser les outils de modularité vus en cours et TP (compilation séparé, fichier makefile, etc.).

Selon la version choisie, vous serez notés sur un maximum de :
- version 1 : 6 points
- version 2 : 8 points
- version 3 : 10 points
- bonus : 1 point (pour un "plus", par exemple : utilisation de git, makefile poussé, solution innovante, etc.)

Les critères de notation sont :
- efficacité du code (efficacité des algorithmes),
- forme du code (modularité, lisibilité, commentaires bien choisis),
- présence d'en-têtes de fonctions et de modules informatifs, conventions de présentation homogènes...),
- présence de fichiers de tests unitaires et de fichiers de tests fonctionnels.
- ergonomie (ie utilisabilité, jouabilité) du programme.

Cette notation sera faite lors de la recette et de l'étude ultérieure du code par le tuteur.

Vous serez donc potentiellement notés sur 21 points. Vous devez toutefois être conscients que les points de 16 à 21 seront plus durs à gagner.

2.2.1 Règles de codage minimales

Quand on programme à plusieurs, il est indispensable de s'organiser. Certaines règles de codage peuvent aider à cela :

- Au niveau des modules : Chaque module de l'application débutera par une en-tête. Vous définirez un modèle d'en-tête dans lequel devront figurer les informations que vous jugerez nécessaires pour une bonne compréhension du rôle de ce module.

  Exemple : le ou les noms des auteurs, les dépendances, la date de dernière modification,...

- Au niveau des sous-programmes : Chacun sera commenté par des commentaires indiquant le rôles des paramètres d'entrée et de sortie ainsi que le domaine des valeurs attendues.

- Les commentaires figurant dans le code devront être pertinents et placés en fonction des différentes étapes de l'algorithme associé.
- Choisissez des règles de nommage des variables en fonction du statut de celles-ci (locales à une fonction, à un module, externes, initiales du programmeur, etc.).

  Exemple : g_ai_fin_jeu = false ; pour dire que la variable appartient au module "ia" et est globale dans ce module.

2.2.2 Les tests et les fichiers de tests

Soyez rigoureux et organisés dans vos tests. Voici un exemple de méthodologie :

1. Tests unitaires : Un fichier de test pour chaque fichier test_u_module contenant un main appelant les diverses fonctions du module à tester.
2. Tests d'intégration au niveau d'un module test_nom_du_module permettant de tester les interactions entre plusieurs sous programmes ou fichiers, du module.
3. Tests fonctionnels : Un fichier test test_fonctionnel qui permet de tester le logiciel intégré avec une jeu de tests ou des grilles prédéfinies.
4. On pourra placer chacun de ces tests dans un sous-répertoire du répertoire courant et créer un Makefile associé à chaque test.

Remarque : l'intérêt de garder les tests est de pouvoir faire des tests régressifs en cas de changement majeur.

3 Annexes

3.1 Sujet : le jeu HEX

Cette partie décrit le principe du jeu et les exigences du client pour les trois versions incrémentales du programme à réaliser. Les aspects techniques (algorithmes, calculs, structures de données) sont décrits dans un document technique séparé que vous trouverez sur Moodle.

3.1.1 Présentation du jeu (https ://fr.wikipedia.org/wiki/Hex)

Le jeu de Hex est un jeu de société combinatoire abstrait pour deux joueurs. Il se joue sur un tablier en forme de losange dont les cases sont hexagonales. Toutes les dimensions du côté du losange sont possibles, la plus traditionnelle est celle composée par 11 hexagones, mais on trouve aussi les valeurs 13 ou 19. L'un des inventeurs du jeu, John Nash, préconise un tablier de taille 14×14. Ce jeu possède des similarités avec le go. Ce jeu, inventé par des mathématiciens fait uniquement appel à la logique, à l'image du go ou des échecs. Son étude est source d'inspiration, non seulement en théorie des jeux, mais aussi pour d'autres branches des mathématiques comme la topologie ou la géométrie algébrique. Si l'on sait qu'il existe une stratégie gagnante pour le premier joueur, cette stratégie est inconnue si le tablier n'est pas de petite taille (de côté strictement plus petit que 9). La recherche de stratégies efficaces, à défaut d'une stratégie gagnante, est l'objet d'études en intelligence artificielle.

3.1.2 But du jeu

Le but du jeu est, pour chaque joueur, de relier les deux bords opposés du plateau de jeu (rouge ou bleu) avec un chemin continu de pions.

3.1.3 Les coups

Chaque joueur joue à tour de rôle en posant un pion sur une case libre n'importe où sur le plateau de jeu. Le premier qui crée un chemin contigu de pions gagne la partie.

3.1.4 Ressources pour le jeu

Youtube : Micmaths, la chaîne de Michaël Launay :

`https://www.youtube.com/playlist?list=PLNefH6S6myiMPc5-9XAeUPfxe21Hxso-s`

Articles et sites : Jeu en ligne :

`http://www.novelgames.com/fr/hex/`
`http://fr.boardgamearena.com/#!join (nécessite une inscription)`

Sites :

`http://hexwiki.amecy.com/index.php/Main_Page`
`https://chessprogramming.wikispaces.com/Hex`

Jeu pour Windows :

`http://vanshel.com/Hexy/`

3.2 Cahier des charges de programmation

Le travail à faire comprend donc le programme de jeu proprement dit et la documentation. Le programme est divisé en trois niveaux et en deux catégories de programmation : la gestion du jeu qui sera en Java et les calculs du jeu qui seront en C.

3.2.1 Version 1 : take it easy ! Humain contre humain (6 points en jeu)

L'interface (Java) :

1. Initialiser la taille du plateau de jeu (défaut 11×11) et l'afficher en ascii (simplification car on pourrait faire une IHM en Java).

```
W W W W W W W/B
 B . * * o * * B
  B o o o * * o B
   B . o * o . . B
    B . * . . b . B
     B * * . * . o B
      B * . . . . . B
       B/W W W W W W W
* pour les noirs (B) et o pour les blancs (W)
```

2. Afficher l'ensemble des fonctionnalités sous forme de « menu » (du texte sur l'écran pour simplifier là aussi),
3. Afficher des informations : joueur, numéro du tour de jeu, dernier coup joué, nombre de coups, etc.,
4. Gérer les actions de l'utilisateur,
5. Lancer les fonctionnalités associées (voir ci-dessous),
6. Mettre à jour "l'interface" : menus, plateau de jeu avec les nouvelles cases jouées contenant les "pions" des joueurs.

Fonctionnalités du jeu (Java et appels de fonction C) :

1. Initialiser le plateau pour le jeu (une taille variable : $N \times N$).
2. Gérer le jeu humain contre humain
3. choix du joueur qui commence.
4. gérer les tours de jeu de chaque joueur.
5. vérifier la légalité des coups des joueurs.
6. détection d'un vainqueur éventuel.
7. Gérer une historique, la sauvegarde, la restauration, l'annulation du dernier coup, l'abandon de la partie en cours, lancer une nouvelle partie en cours de partie.
8. Quitter le jeu avec ou sans sauvegarde.

Contenu technique du jeu (en C) :

Voir la partie $V1$ du document technique.

3.2.2 Version 2 : get it tough ! Humain contre ordinateur, la tentation de la force brute (2 points de plus en jeu)

Programmer la partie $V2$ du document technique et l'intégrer à votre programme (C et Java).

3.2.3 Version 3 : dirty badass fighting ! Humain contre ordinateur, let's fight ! Les outils du combat (2 points de plus en jeu)

Programmer la partie $V3$ du document technique (C et Java). Cette partie est plus créative.

3.3 Ressources

3.3.1 Moodle

Vous y trouverez :
- Le document technique détaillant les méthodes à utiliser pour les différentes versions du programme.
- L'échéancier.
- Des modèles de documents.
- Le forum.

3.3.2 Votre tuteur

Il est conseillé de préparer des questions précises avant chaque réunion.

3.4 Fichiers de configuration et sauvegarde

Les fichiers de configuration sont des fichiers texte (.txt) qui contiennent les informations suivantes :
Le fichier commence par le mot clé

```
\hex
```

suivi de :

```
\dim n
```

où n est la dimension du plateau de jeu. La description du plateau de jeu annoncé par le mot clé :

```
\board
```

puis viennent n lignes de n caractères représentant une configuration du plateau de jeu :

- Si la case (i, j) est occupée par un pion noir, le caractère sera « * »
- Si la case (i, j) est occupée par un pion blanc, le caractère sera « o »
- Si la case (i, j) n'est pas occupée le caractère sera « . »

Le mot clé

```
\endboard
```

termine cette description.

Le fichier est clos par le mot clé

```
\endhex
```

Le programme doit vérifier la cohérence de toutes les informations du fichiers par rapport aux règles.

Exemple : Le plateau

```
W W W W W W W/B
 B . * * o * * B
  B o o o * * o B
   B . o * o . . B
    B . * . . b . B
     B * * . * . o B
      B * . . . . . B
       B/W W W W W W W
* pour les noirs (B) et o pour les blancs (W)
```

sera représentée par les lignes :

```
\hex
\dim 6
\board
. * * o * *
o o o * * o
. o * o . .
. * . . b .
* * . * . o
* . . . . .
\endboard
\endhex
```

Pour la mémorisation de l'historique du jeu on ajoute avant le

```
\endhex
```

une nouvelle section :

```
\game
```

suivie d'une ligne par coup :

```
\play joueur i j
```

```
Exemple : \play * 5 5
```

On termine par un

```
\endgame
\endhex
```

A la fin du fichier on doit pouvoir déduire le joueur dont c'est le tour.

Le programme devra vérifier que les coups sont cohérents et construisent bien le plateau.

4 Etudes diverses liées au projet

Ces travaux ne seront pas notés dans la partie projet mais seront notés comme des TP des matières correspondantes et la note intégrée au CC.

4.1 Structures de données

La partie C du code sera notée par les enseignants de l'UE Structures de Données, et la note intégrée aux MCC de cette UE.

4.2 Programmation orientée objets

Le code Java du programme sera notée par les enseignants de l'UE correspondante, et la note intégrée aux MCC de cette UE.

4.3 Bases de données

Le travail dans cette partie sera notée par les enseignants de l'UE BD.

On veut mettre en place une bibliothèque (base de données) de parties. La bibliothèque sauvegardera les informations concernant les joueurs (pseudo, email et année de naissance) et les parties.

Chaque partie implique deux joueurs humains ou un joueur humain et un joueur ordinateur.

Pour les joueurs ordinateurs l'email sera celui du concepteur du programme simulant le joueur et l'année de naissance sera la date de création du programme.

Chaque partie est constituée des listes de coups joués par le joueur 1 et par le joueur 2.

Cette base de données devra permettre de :

- retrouver les parties effectuées par un joueur à une date donnée et aussi toutes les parties effectuées par un joueur.
- retrouver les parties gagnées par un joueur
- retrouver les parties abandonnées auxquelles à participé un joueur
- retrouver les joueurs rencontrés par un joueur donné
- retrouver les parties débutée par un coup donné

Donner les différents modèles et requêtes permettant de mettre en place une telle base de données à l'aide du SGBD Oracle, tout en laissant la possibilité de l'implémenter dans un autre environnement de bases de données.