La bataille du café

1 Préambule

Ce projet tutoré est en grande partie inspiré de l'épreuve de programmation de l'édition 2019 des 24 heures des IUT Informatique, qui s'est déroulée les 25 et 26 mai dernier à l'IUT du Havre.

Il mêle des concepts de programmation système (communication réseau par l'intermédiaire des sockets, d'où son intersection avec le module M3101 du DUT Informatique) et surtout d'intelligence artificielle (il s'agit de jouer plus malin que l'adversaire).

Ce projet est réparti sur les semestres 3 et 4 du DUT Informatique. Il est à réaliser en quadrinôme par l'intermédiaire du langage C#.

Restez calme. Relisez ce préambule.

2 Le scénario à modéliser

Il s'agit de modéliser le comportement d'un planteur de café qui découvre un îlot inhabité et inexploré. Ce planteur tente de prendre le contrôle d'un maximum de parcelles de cet l'îlot afin d'y semer des graines de café. Bien entendu, un adversaire farouche et déterminé essaie lui aussi de contrôler des parcelles de l'îlot.

Une guerre sans merci s'engage donc entre vous (le planteur) et un serveur informatique (l'adversaire). Le gagnant obtient le contrôle de l'îlot.

2.1 La structure d'un îlot

Un îlot est toujours constitué de dix-sept **parcelles**.

Chaque parcelle est constituée de plusieurs zones carrées appelées **unité**. On donne figure 1 des exemples de parcelles de taille 4.

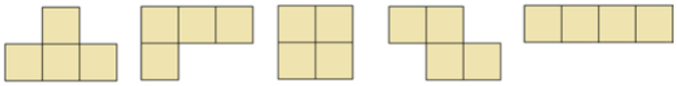

FIGURE 1 – Quelques exemples de parcelles de taille 4

Un îlot possède toujours :
- 4 parcelles de 6 unités
- 5 parcelles de 4 unités
- 4 parcelles de 3 unités
- 4 parcelles de 2 unités

→ Ceci fait un total de 64 unités réparties sur 17 parcelles.

Un îlot peut également contenir des blocs de type **Mer** ou **Forêt**, composés d'unités sur lesquelles il est impossible de planter des graines de café. Un bloc **Mer** peut se situer à l'intérieur de l'îlot, auquel cas il

s'agit d'un lac.

Un îlot s'inscrit toujours dans une grille de taille 10 unités × 10 unités.

On donne figure 2 un exemple de carte valide.

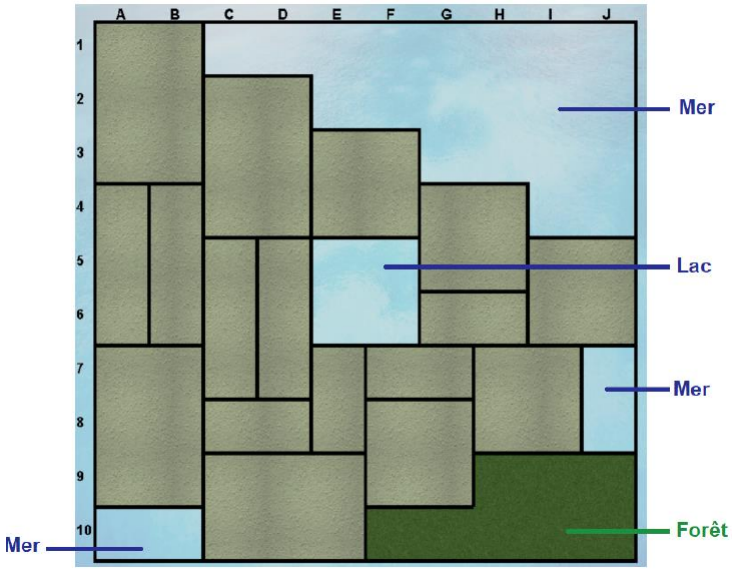

FIGURE 2 – Un exemple d'îlot

2.2 Contrôle des parcelles et décompte des points

Chaque planteur dispose de 28 graines.

A tour de rôle, les planteurs placent une graine sur une unité vierge (une unité ne peut recevoir qu'une seule graine ; il est donc impossible de planter une graine sur une unité déjà occupée). Un planteur contrôle une parcelle quand il y possède plus de graines que son adversaire.

Le nombre de points apportés par le contrôle d'une parcelle dépend de la taille de la parcelle.

Taille de la parcelle	**Points gagnés**
6 unités	6
4 unités	4
3 unités	3
2 unités	2

Par exemple, si la partie devait s'arrêter sur la configuration des plantations représentée par la figure 3 :
- Le planteur de graines blanches contrôle 1 parcelle de 2 unités et 1 parcelle de 3 unités : cela lui rapporte $2 + 3 = 5$ points
- Le planteur de graines noires contrôle 3 parcelles de 4 unités : cela lui rapporte $3 \times 4 = 12$ points

A ces points de contrôle des parcelles s'ajoute un bonus lié à la plus grande étendue de graines adjacentes orthogonalement : x graines adjacentes rapportent x points supplémentaires en fin de partie (les frontières des parcelles ne sont pas prises en compte).

Par exemple, si la partie devait s'arrêter sur la configuration déjà étudiée (figure 4) :

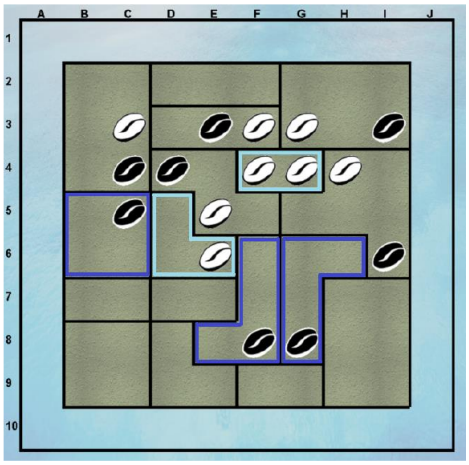

FIGURE 3 – Exemples de décompte des points de contrôle de parcelles

— La plus grande étendue pour le planteur de graines blanches est de 5 unités : ce joueur reçoit un bonus de 5 points
— La plus grande étendue pour le planteur de graines noires est de 3 unités : ce joueur reçoit un bonus de 3 points

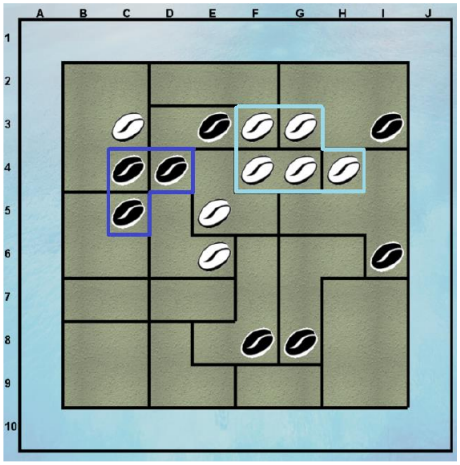

FIGURE 4 – Exemples de décompte des points des plus grandes étendues

2.3 Règles de pose des graines

— **R1**. La première pose est libre.
— **R2**. Il est interdit de placer une graine sur une unité de type Mer ou Forêt.
— **R3**. Un joueur doit poser une graine sur la même ligne ou la même colonne que la dernière graine placée.
— **R4**. Un planteur ne peut pas planter une graine sur la même parcelle que la dernière graine posée.
— **R5**. En cas de pose invalide, le planteur perd sa graine et son tour.
— **R6**. Si aucune des cases disponibles ne satisfait les règles **R3** et **R4**, alors la partie est terminée.
— **R7**. Dès qu'un planteur a utilisé ses 28 graines, la partie est également terminée.

Dès que la partie est terminée, le nombre de points de chaque planteur est déterminé (points de contrôle des parcelles auquel s'ajoute le bonus lié à la plus grande étendue).

3 Dialogue avec le serveur

3.1 Mécanisme de communication

Le dialogue entre le serveur et un joueur est réalisé par l'intermédiaire d'une socket à travers le protocole TCP.

3.2 Etape 1 : envoi de la carte au joueur

Sitôt connecté, le client reçoit de la part du serveur la carte de l'îlot.

Chaque unité est représentée par un nombre entier compris entre 0 et 15. Si ce nombre vaut zéro, cela signifie que l'unité ne possède aucune frontière avec ses quatre voisines. Sinon ce nombre est une combinaison de puissances de 2 et chaque puissance indique qu'il existe une frontière sur un des cotés de l'unité (figure 5) :

- 2^0 : il y a une frontière au nord
- 2^1 : il y a une frontière à l'ouest
- 2^2 : il y a une frontière au sud
- 2^3 : il y a une frontière à l'est

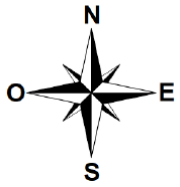

FIGURE 5 – Les quatres points cardinaux

Par exemple, une unité représentée par le nombre 9 possède une frontière au nord et à l'est car $9 = 2^3+2^0$.

Pour différencier les unités de type Mer ou Forêt, pour lesquelles la règle ci-dessus s'applique, on ajoute respectivement 64 et 32 à la combinaison des puissances de 2. Par exemple, une unité représentée par le nombre 70 est de type **Mer** et comporte une frontière à l'ouest et au sud car $70 = 64 + 2^1 + 2^2$.

Ces nombres sont contenus dans une trame dans laquelle ils sont séparés par le caractère « : » tandis que le caractère « | » introduit une nouvelle ligne de la carte.

Ainsi, la trame suivante est correcte et représente la carte montrée figure 6 (chaque lettre minuscule identifie une parcelle).

```
3:9:71:69:65:65:65:65:65:73|2:8:3:9:70:68:64:64:64:72|6:12:2:8:3:9:70:68:64:72|11:11:6:
12:6:12:3:9:70:76|10:10:11:11:67:73:6:12:3:9|14:14:10:10:70:76:7:13:6:12|3:9:14:14:11:7:
13:3:9:75|2:8:7:13:14:3:9:6:12:78|6:12:3:1:9:6:12:35:33:41|71:77:6:4:12:39:37:36:36:44|
```

3.3 Etape 2 : placement des graines

Dès que le client a reçu la carte, les étapes suivantes sont mises en oeuvre :

1. il joue en premier en envoyant au serveur la chaine de caractères `A:xy` : ceci signifie que le client place une graine sur l'unité située à la ligne x et à la colonne y. Ces coordonnées x et y sont basées

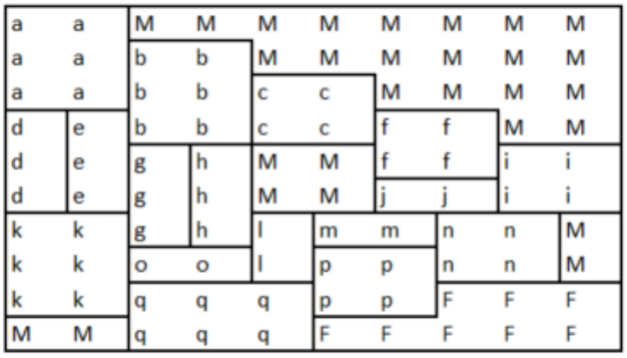

FIGURE 6 – La carte correspondant à la trame donnée en exemple.

zéro. Ainsi, `A:25` signifie que le client veut planter une graine sur l'unité située à la *troisième* ligne et à la *sixième* colonne

2. A cette proposition, le serveur répond par la chaine de caractères `INVA` si le coup du client est invalide ou par `VALI` si le coup est correct.
3. Ensuite, le serveur joue à son tour en envoyant au client `B:xy` : ceci signifie qu'il place une graine à la ligne x et à la colonne y. Si le serveur ne peut pas jouer parce qu'aucune unité n'est disponible, il envoie `FINI` : la partie est terminée
4. Enfin, le serveur envoie `ENCO` si le client peut encore jouer ou `FINI` s'il n'y a plus d'unités possibles pour que le client puisse placer une graine ou si le nombre limite de coups est atteint, auquel cas la partie s'arrête.
5. Après avoir envoyé `FINI`, le serveur envoie le score au client, sous la forme d'une chaine de caractères de longueur 7 `S:aa:bb`, où :
   (a) *aa* est le score final du client
   (b) *bb* est le score final du serveur
6. Si la partie n'est pas terminée, retour en 1.

Le schéma suivant résume les échanges de jeux :

```
CLIENT                              SERVEUR
---------------- A:xy ------------------>
<---------- VALI ou INVA ---------------
<-----------B:xy ou FINI ---------------
<---------- ENCO ou FINI ---------------

---------------- A:xy ------------------>
...
<-------------- FINI -------------------
<------------- S:aa:bb -----------------
```

4 Contraintes de programmation et évaluation

4.1 La programmation

Le projet est réalisé en quadrinôme. Le client est programmé en C#, indifféremment sous Microsoft Windows (framework .Net et Visual Studio) ou sous Unix (à travers *mono* et *MonoDevelop*).

Il est convient bien entendu de réaliser un implémentation orientée objet.

Le serveur sera installé sur un serveur de notre département. Il tournera en permanence, vous permettant de tester à tout instant votre client.

- Durant le semestre 3 tournera une **version simplifiée** du serveur, dont l'action sera uniquement de fournir la carte de l'îlot au client.
- Durant le semestre 4, un serveur **complet** sera mis en place : il vous permettra de mener une véritable partie contre le serveur

4.2 Qualité du code

4.2.1 Structuration

- Le logiciel doit être structuré en utilisant les techniques de programmation orientée objets.
- Le rôle de chaque classe doit être clairement identifié.

4.2.2 Norme de codage

- Le code doit être correctement indenté
- Un sous programme ne devrait pas contenir plus de quinze lignes de code.
- On respectera le principe DRY (*don't repeat yourself*) : la duplication de code est proscrite.

4.2.3 Explications et commentaires

- Tous les identificateurs doivent avoir un nom explicite qui permet de comprendre leur utilité. En C#, il est d'usage de suivre la convention *Camel Case* : les différents mots constituant un identificateur commencent par une majuscule et ne comportent pas de caractère de séparation. De plus, pour les identificateurs de variables, le premier mot commence par une minuscule. Par exemple : *int largeurPixel ;*
- Privilégier la lisibilité des noms de variables par rapport à la concision. Par exemple *prixHorsTaxe* est préférable à *pxHT*
- Toutes les méthodes doivent être documentées : rôle de la méthode, description des paramètres, description de la valeur de retour
- Par convention, les identificateurs de méthodes en C# suivent également la norme *Camel Case* mais, contrairement aux identificateurs de variable, commencent leur premier mot par une majuscule (exemple : *DecodageCarte*).

4.3 ENT

Un cours à propos de ce projet tutoré sera ouvert sur l'ENT de l'IUT `ent.iut-amiens.fr`. On pourra y trouver :

- des exemples de carte, qui vous permettrons de décoder des cartes hors ligne (sans se connecter au serveur)
- un support de cours sur les sockets en C#
- des renseignements supplémentaires
- les dates cruciales liées au projet

Cette communication via l'ENT sera doublée d'une communication par mail : soyez vigilants sur votre mail universitaire !

4.4 L'évaluation

4.4.1 Semestre 3

Il s'agira de fournir un rapport en format papier qui contiendra :
- une page de garde complète
- une analyse et une décomposition en sous problèmes du travail à réaliser durant le semestre 3 (connexion au serveur et décodage de la carte)
- le code d'un client qui se connecte au serveur, récupère l'îlot, le décode puis l'affiche (imprimé en police à chasse fixe de type `Courrier` et en taille 8)
- une première proposition d'une modélisation objet du problème, à travers le formalisme de votre choix

4.4.2 Semestre 4

Il s'agira de fournir :
- le code complet du client, téléversé sur l'ENT
- un rapport qui comportera :
  - une page de garde complète
  - une introduction
  - la modélisation objet complète du problème
  - une explication de la stratégie utilisée pour combattre l'adversaire
  - une conclusion, qui intégrera les apports de ce projet, les difficultés rencontrées et les éventuels axes d'amélioration du programme
  - un lien vers le dépôt *git* du projet ou, sinon, en annexe, le code source du programme, en police à chasse fixe de type `Courrier` et en taille 8

En outre, la note finale intégrera les performances du client à travers un tournoi qui mettra en compétition toutes les IA. Une phase de sélection organisée au sein de chaque groupe permettra de dégager les cinq finalistes qui s'affronteront lors du tournoi suprême, au terme duquel sera couronné le quadrinôme gagnant.