Projet: Stratego
Licence 2 Informatique

Julien Bernard

2015

Ce document présente le projet conjoint Algorithmique et Système. Le sujet étant complexe et couvrant deux UE, il se peut qu'il soit imprécis ou incomplet par endroit. Dans ce cas, vous signalerez à votre encadrant les manques et une correction sera apportée à ce document.

Stratego

Les règles du jeu

Le Stratego [1] se joue à 2 joueurs sur un plateau carré de 92 cases (10 cases de côté moins 2 lacs carrés de 4 cases chacun). Chaque joueur possède 40 pièces.

Les pièces représentent des unités militaires et ont deux faces. Une face ne peut être vue que par un seul joueur à la fois, l'autre ne voyant que la couleur de la pièce. Les pièces sont placées de telle façon que le joueur ne voit que le rang de ses propres pièces.

Au début de la partie chaque joueur dispose ses pièces comme il l'entend sur ses quatre premières rangées. Cette pré-phase du jeu est stratégique et déterminante pour la suite de la partie.

Chaque joueur déplace une pièce d'une case par tour : à gauche, à droite, en avant ou en arrière (pas en diagonale). Une attaque se produit quand le joueur déplace sa pièce sur une case déjà occupée par l'adversaire. Chaque joueur montre alors sa pièce à l'adversaire. La pièce la plus forte reste en jeu, l'autre est éliminée ; en cas d'égalité, les deux sont éliminées.

Voici les pièces classées de la plus forte à la plus faible (la force entre parenthèses) :

— le Maréchal (10), 1 par joueur
— le Général (9), 1 par joueur
— les Colonels (8), 2 par joueur
— les Commandants (7), 3 par joueur
— les Capitaines (6), 4 par joueur
— les Lieutenants (5), 4 par joueur
— les Sergents (4), 4 par joueur
— les Démineurs (3), 5 par joueur
— les Éclaireurs (2), 8 par joueur
— l'Espion (1), 1 par joueur

1. Ces règles du jeu sont tirées de Wikipédia

Pièce	Valeur
Maréchal	1
Général	2
Colonel	3
Commandant	4
Capitaine	5
Lieutenant	6
Sergent	7
Démineur	8
Éclaireur	9
Espion	10
Drapeau	11
Bombe	12

TABLE 1 – La représentation des pièces

— le Drapeau (0), 1 par joueur

À ces pièces s'ajoutent les Bombes (6 par joueur). Ni les Bombes ni le Drapeau ne se déplacent.

Le **but du jeu** est de capturer le Drapeau de l'adversaire ou d'éliminer assez de pièces adverses afin que l'adversaire ne puisse plus faire de déplacement.

Certaines pièces obéissent à des règles spéciales :

— Si l'Espion, grade le plus faible, attaque le Maréchal, grade le plus élevé, l'Espion gagne (si le Maréchal attaque en premier, le Maréchal gagne) ;

— Toute pièce attaquant une Bombe est éliminée, sauf le Démineur qui prend alors la place de la Bombe (si une pièce autre qu'un Démineur attaque une Bombe, cette pièce est éliminée, et la Bombe reste en place jusqu'à l'éventuelle attaque d'un Démineur) ;

— L'Éclaireur peut se déplacer d'autant de cases libres qu'il le souhaite, en ligne droite.

But du projet

Le but du projet est d'implémenter un joueur de Stratego et un arbitre capable de faire jouer deux joueurs. Les échanges entre l'arbitre et les joueurs passeront par les flux standards. Plus précisément, le joueur recevra les paramètres sur son entrée standard et, à chaque tour, devra rendre une réponse sur sa sortie standard.

Chaque joueur doit rendre une réponse en moins de 0,5 seconde.

Conventions

La table 1 donne la représentation des pièces qui sera utilisée lors des échanges. Chaque pièce est représentée par un entier non-nul. Cet entier est indépendant de sa force ou du nombre de pièces disponibles.

La figure 1 montre la disposition des pièces sur le plateau au démarrage. Une position sur le plateau est représentée par un couple constitué d'un nombre qui représente l'abscisse et une lettre qui représente l'ordonnée. La position A1 est située en haut à gauche du plateau. Les cases C4, C5, D4, D5, G4, G5, H4, H5

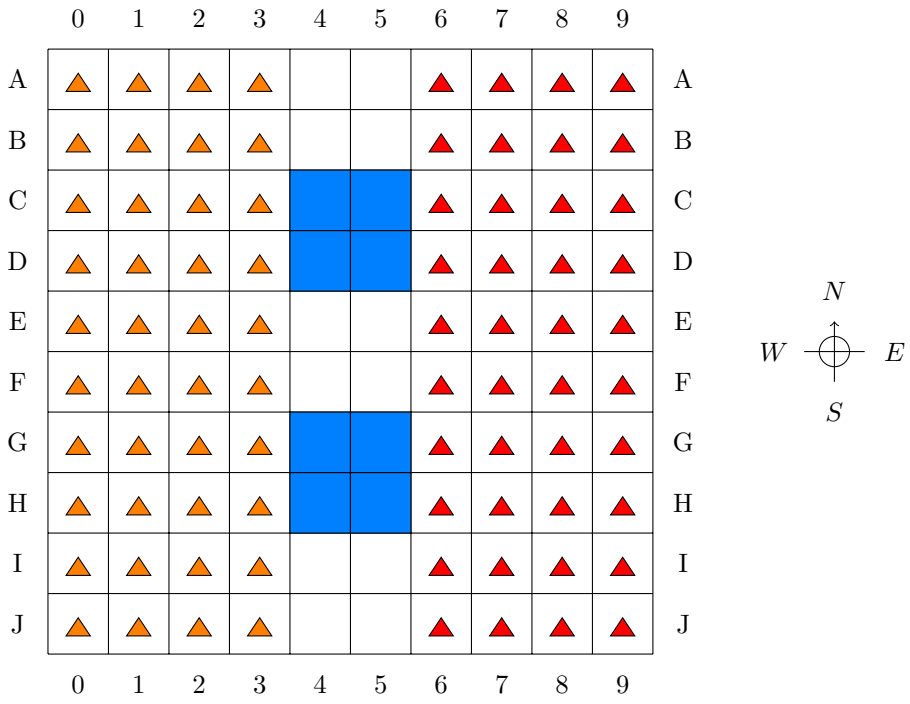

FIGURE 1 – La disposition des pièces sur le plateau de Stratego au démarrage

Direction	Valeur
Nord	`'N'`
Sud	`'S'`
Est	`'E'`
Ouest	`'W'`

TABLE 2 – La représentation des directions

sont inaccessibles. Le joueur 0 se situe à gauche du plateau (pièces oranges), le joueur 1 se situe à droite du plateau (pièces rouges).

La table 2 montre la représentation des directions. Chaque direction est représentée par un caractère. L'orientation du plateau est indiquée sur la figure 1.

Les échanges

Au démarrage, le joueur reçoit sur son entrée standard une ligne contenant son numéro (0 ou 1). Puis, il doit renvoyer sur sa sortie standard la liste des 40 pièces, une par ligne, qu'il a disposées sur le plateau, en commençant par la ligne A, de gauche à droite, puis la ligne B, de gauche à droite, etc. L'arbitre envoie alors une chaîne de caractères sur une ligne : `OK` si la configuration est acceptée, `KO` si la configuration n'est pas acceptée. Les causes possibles sont : le nombre de pièces de chaque type ne correspond pas aux règles du jeu, un nombre reçu ne correspond à aucune pièce.

Puis, dans une boucle infinie, le joueur 0 envoie son coup à l'arbitre. L'arbitre

envoie le résultat du coup. Puis le joueur 0 attend le coup adverse et le résultat du coup adverse. Le joueur 1 attend le coup adverse et le résultat du coup adverse. Puis il envoie son coup et l'arbitre lui envoie le résultat du coup.

Pour envoyer un coup, le joueur envoie les coordonnées de la pièce qu'il souhaite déplacer, sur une première ligne, puis la direction dans laquelle il souhaite qu'elle se déplace, sur une seconde ligne et le nombre de cases dont elle doit se déplacer, sur une troisième ligne.

Pour envoyer le résultat d'un coup, l'arbitre envoie un code d'un caractère sur une ligne et des informations supplémentaires le cas échéant :

- le code `N` *(Nothing)* signifie que le coup est valide et que la case d'arrivée est vide. Rien de plus ne se passe.
- le code `B` *(Battle)* signifie qu'il y a une bataille entre deux pièces. L'arbitre envoie alors la valeur de la pièce qui attaque sur une ligne puis la valeur de la pièce qui est attaquée sur une autre ligne, ce qui permet à chaque joueur de déterminer le résultat.
- le code `E` *(Error)* signifie qu'il y a eu une erreur dans le coup joué. Les causes possibles sont : la position indiquée ne contenait pas de pièce du joueur ; la direction indiquée ne permettait pas au joueur de se déplacer ; la pièce indiquée ne peut normalement pas bouger. Une erreur arrête immédiatement la partie et celui qui a fait l'erreur perd.
- le code `F` *(Failure)* indique que le joueur adverse a subi une panne. Les causes possibles sont : le programme du joueur adverse s'est arrêté de manière imprévue ; le programme du joueur adverse a mis trop de temps à répondre. Une panne arrête immédiatement la partie et celui qui est en panne perd.

Partie 1 : Algorithmique, le joueur

Cette partie sera évaluée dans l'UE «Algorithmique et structures de données».

Le joueur doit définir une stratégie initiale pour placer ses pièces. Il y a plusieurs manières de faire. Il est possible de définir une stratégie initiale unique, mais il est également possible de définir une stratégie initiale dynamiquement, soit totalement au hasard (pas recommandé), soit parmi une liste de stratégies initiales, soit avec des schémas de placement de certaines pièces et un peu de hasard. Dans tous les cas, il faut veiller à bien placer ses pièces en fonction du numéro de joueur attribué, et éviter d'écrire deux versions des fonctions (une pour chaque numéro de joueur).

Le joueur doit ensuite définir un coup à jouer. Là encore, il est possible d'adopter plusieurs stratégies. La base consiste à faire la liste des coups jouables, et parmi ces coups, choisir celui qui semble le meilleur. Le choix le plus simple consiste à tirer au hasard, mais ce choix n'est pas forcément le plus efficace. À partir de là, il faut déterminer des fonctions qui permettront d'évaluer une position et donc de savoir la bonne tactique à adopter. Ces fonctions peuvent être très naïves mais peuvent également être très complexes (par exemple, si elles prennent en compte plusieurs coups à l'avance).

Quoiqu'il en soit, il vous faudra également maintenir l'état du jeu, c'est-à-dire faire bouger les pièces sur votre version du plateau et résoudre les combats entre pièces. Vous pourrez également garder en mémoire les pièces que votre

adversaire aura révélées par une attaque de manière à utiliser cette information dans vos stratégies. La mise à jour du plateau est essentielle, notamment pour ne pas risquer d'être éliminé à cause d'un mouvement interdit.

Enfin, il faudra éventuellement gérer la limite de 0,5s pour répondre. Si vos algorithmes sont assez rapides, alors aucune gestion n'est nécessaire. En revanche, si votre stratégie est assez complexe, il peut être intéressant de mettre en place des mécanismes pour arrêter la recherche et rendre un résultat dans le temps imparti.

Évaluation

Vous serez évalués sur le choix des structures et des algorithmes utilisés, en particulier leur complexité. Il vous est demandé de commenter votre code *abondamment* de manière à expliquer vos choix.

Un tournoi aura lieu permettant de tester vos algorithmes les uns contre les autres. Les trois premiers du tournoi recevront un point bonus sur leur projet pour la partie Algorithmique. L'arbitre utilisé pendant le tournoi sera mis à disposition avant le tournoi par les encadrants. La date et les modalités exactes du tournoi seront précisés ultérieurement.

Partie 2 : Système, l'arbitre

Cette partie sera évaluée dans l'UE «Système et programmation système».

Le rôle de l'arbitre est de coordonner l'ensemble du jeu. Pour cela, l'arbitre gère toute la durée de vie des deux processus joueurs. Il est notamment en charge du lancement des processus et de la mise en place des moyens de communication entre lui et les joueurs qui permettront en particulier aux joueurs d'annoncer leurs coups et à l'arbitre d'annoncer le résultat. Il sera donc nécessaire de mettre en place des tubes et de les gérer convenablement.

La lecture et l'écriture des données doivent faire l'objet d'une attention très particulière. En effet, les données passent sous forme textuelle mais l'API utilisée pour les manipuler (les appels systèmes) n'est pas forcément adaptée à l'usage (pas de `fgets(3)`, pas de `fprintf(3)`). Il sera sans doute nécessaire de réimplémenter des fonctionnalités qu'on trouve dans l'API de plus haut niveau pour faciliter les échanges entre l'arbitre et les joueurs.

Pendant le jeu, de nombreuses situations critiques peuvent se produire. Le joueur peut s'arrêter brusquement en cours de partie, il peut ne pas répondre ou mettre trop de temps à répondre. Vous devrez prendre en charge ces situations de manière adéquate (notamment avec des gestionnaires de signaux). N'hésitez pas à utiliser des versions avancées des mécanismes vus en cours (indice : `sigaction`) de manière à avoir un contrôle assez fin des processus joueurs.

Un des challenges du projet est de limiter le temps de chaque joueur à 0,5s. Vous devrez trouver et implémenter une solution qui permette non seulement de détecter qu'un joueur a mis plus de 0,5s, mais aussi de gérer le cas où le joueur ne répond pas (par exemple parce qu'il est pris dans une boucle infinie). Il faut que l'arbitre puisse continuer à fonctionner dans tous les cas et détecte tous ces cas. La limite de 0,5s est une limite garantie pour le joueur, c'est-à-dire qu'il se peut qu'il puisse avoir plus de 0,5s mais jamais moins de 0,5s.

À la fin de la partie, vous devrez ordonner aux joueurs de terminer. Vous ne pouvez faire aucune supposition sur les joueurs, il faudra donc implémenter un mécanisme suffisamment générique pour qu'il puisse être utilisé avec n'importe quelle implémentation respectant l'énoncé.

Un des rôles de l'arbitre est de vérifier que personne ne triche. Vous pourrez ajouter du code de vérification mais vous ne serez *pas* évalué sur cette partie. Vous serez évalué uniquement sur les aspects système. Mais il est évident que l'arbitre devra avoir une connaissance minimale du jeu et de son déroulement. En particulier, votre arbitre devra gérer obligatoirement les résultats de coup de type *Nothing*, *Battle* et *Failure*. La gestion des résultats de coup de type *Error* est facultative (mais fortement recommandée pour pouvoir tester votre joueur).

L'arbitre prend en paramètre les deux programmes des deux joueurs.

De manière à ce que les affichages des joueurs ne perturbent pas l'affichage de l'arbitre, l'arbitre doit récupérer l'erreur standard des deux joueurs et la placer dans des fichiers de log nommés `player0.log` et `player1.log`.

Évaluation

Vous serez évalués sur la correction et la qualité de votre implémentation, notamment dans la prise en compte des situations critiques. Il vous est demandé de commenter votre code *abondamment* de manière à expliquer vos choix.

Votre arbitre sera notamment testé avec un joueur spécial qui évaluera différents cas critiques.

Consignes générales

Le travail sera à faire et à rendre en binôme.

Vous réaliserez deux programmes, que vous nommerez `player-nom1-nom2` et `referee-nom1-nom2`, correspondant respectivement au joueur et à l'arbitre, où vous remplacerez `nom1` et `nom2` par les noms du binôme. Vous fournirez une archive `tar.gz` avec un répertoire appelé `stratego-nom1-nom2` contenant vos sources et un Makefile pour créer les deux programmes. Il est conseillé de mutualiser tout ce que vous pourrez entre les deux programmes.

Vous utiliserez de l'anglais pour tout votre code, mais les commentaires pourront être en français.

La qualité des programmes rendus sera également évaluée par l'absence de fuites mémoire, et l'utilisation correcte de la mémoire (indice : Valgrind). La pertinence du nommage des structures, des fonctions et des variables sera également appréciée.

Le non-respect des consignes sera sanctionné.