SAÉ 1.1 et 1.2, Implémentation d'un besoin client.

Departement informatique, IUT Grand Ouest Normandie, campus d'Ifs

21 novembre 2024

1 Présentation générale du projet

L'objectif de se projet est d'implémenter un jeu dont les règles sont expliquées ci-dessous.

Il s'agit d'un jeu de pure stratégie à deux joueurs. Les règles du jeu sont présentées de façon détaillées dans la Section 2.

Pour le premier sprint, un moteur implémentant les règles du jeu est fourni, et vous implémenterez une interface utilisateur la plus ergonomique possible. Vous devrez aussi rédiger une notice d'utilisation en anglais au cours de la semaine qui suit. Les éléments de cette phase sont détaillés dans la section 3. Puis, au cours du deuxième sprint, vous devrez implémenter vous même le moteur de jeu.

2 Règles du jeu détaillées

2.1 Règles initiales

Ce jeu se joue à deux joueurs sur un plateau formé de *cellules*, organisé initialement selon une grille de 8 lignes et 7 colonnes.

Nous appelerons les joueurs Nord et Sud en fonction de leur position autour du plateau. Chaque joueur dispose d'une unique pièce de jeu, initialement positionnée sur la cellule située au milieu de la ligne la plus proche du joueur.

À son tour, le joueur doit exécuter deux actions, à savoir déplacer sa pièce puis tuer une cellule, c'est à dire la retirer du plateau. Tout d'abord, le joueur doit déplacer sa pièce sur une cellule adjacente à sa position actuelle, parmi les cellules en contact par une arête ou un point avec sa cellule actuelle. Sur le plateau de départ, la plupart des cellules ont donc 8 cellules adjacentes. Il n'est pas possible de laisser la pièce immobile. Dans un deuxième temps, le joueur doit tuer une cellule. Pour cela, il choisit une cellule du plateau et la tue. Cette cellule doit être une cellule vivante et inoccupée. Une fois la cellule tuée, il ne sera plus possible pour aucun des deux joueurs de se déplacer dessus, ni de l'éliminer à nouveau.

À l'issue de ces deux actions, le joueur adversaire exécute à son tour ces deux actions, et le tour alterne ainsi tant que les deux joueurs sont en mesure de le faire, le nombre de cellule du plateau allant diminuant à chaque tour.

Lorsqu'un joueur est dans l'incapacité de déplacer sa pièce, il a perdu, son adversaire est alors déclaré vainqueur. On notera qu'il est toujours possible d'éliminer une cellule après s'être déplacé, puisque la cellule que le joueur vient de quitter peut être tuée.

2.2 Variantes

Deux variantes aux règles seront proposées en plus des règles classiques. La première consiste simplement à un changement de forme du plateau, les cellules auront alors une forme hexagonale, la plupart des cellules auront donc 6 cellules voisines. La forme générale du plateau sera alors un grand hexagone de 5 cellules de côté, contenant donc 61 cellules. Les autres règles sont inchangées.

La deuxième variante consiste à définir une portée pour l'action consistant à tuer une cellule. Chaque joueur sera en mesure de tuer une cellule uniquement s'il pourrait l'atteindre en au plus 3 déplacements (si la cellule est à au plus 3 pas de la case d'arrivée de sa pièce).

Il est bien sûr possible de combiner les deux variantes pour proposer un jeu sur une grille hexagonale avec une portée limitée.

Bien entendu, votre priorité doit rester de rendre fonctionnel le jeu selon les règles initiales.

3 Première phase, interface utilisateur

Pour une première phase, nous vous fournissons une implémentation d'un module gérant la modélisation du plateau et la bonne application des règles du jeu. Ce module vous est fourni sous la forme d'un fichier précompilé `board.o` accompagné du fichier d'en-tête correspondant `board.h`.

Bien que le `board.o` ne soit pas lisible, vous trouverez une documentation détaillée des fonctions mises à disposition dans le fichier d'en-tête, qui a permis de générer automatiquement une documentation du projet au format `html` ou LaTeX. La documentation ainsi générée est accessible en ligne à l'adresse `https://dorbec.users.greyc.fr/SAE` ou sur ecampus.

Il est recommandé de bien prendre le temps de noter les différentes fonctions proposées, afin de ne pas perdre de temps à essayer de gérer des fonctionnalités déjà implémentées.

À l'issue de cette première phase, vous devez rendre deux documents

3.1 Interface de jeu en console

Dans un premier temps, vous chercherez à faire une interface dans la console permettant de jouer une partie à deux joueurs, alternant dans leurs actions dans une même console. Vous devrez utiliser la bibliothèque *board.o* fournie. Vous pouvez choisir la présentation du jeu selon votre préférence, sans nécessairement suivre l'exemple fourni qui a pour seul objectif de vous aider à comprendre le jeu et les consignes.

Votre programme sera rendu **individuellement** dans l'interface fournie sur ecampus. Vous rendrez une archive `.zip` contenant le ou les fichiers nécessaire à la compilation et l'exécution de votre programme, à l'exception des fichiers *board.h* et *board.o* qui vous sont fournis et n'ont pas pu être modifiés. Vous n'inclurez pas les fichiers précompilés dont vous fournissez le code source.

Vous pouvez travailler en binômes, et si c'est le cas, vous indiquerez dans l'interface *choix de groupe* sur ecampus votre binôme, afin d'éviter que les similarité de code ne soient prises

pour du plagiat. Vous indiquerez aussi explicitement en commentaire dans l'en-tête de votre programme la composition du binôme.

Si des similarités trop importantes persistent entre le code rendu par des étudiants hors binôme (et en particulier entre plus de deux étudiants), cela sera considéré comme de la fraude. Ceci peut bien sûr être aussi causé par un usage d'interface de génération de code (IA).

La facilité de lecture du code, avec notamment un découpage pertinent en fonctions, seront pris en compte pour l'évaluation.

Points d'amélioration possibles Parmi les points d'attention ergonomiques qui sont suggérés dans votre implémentation :

— une interface agréable.

— une protection contre les erreurs de frappes (e.g. si je saisis une lettre quand on me demande un chiffre, je peux continuer la partie).

— le jeu permet de saisir un nom à chaque joueur, et s'en sert dans les affichages.

— le jeu m'aide à identifier les déplacements possibles.

Évaluation croisée À l'issue du sprint, vous recevrez le code d'autres étudiants pour en réaliser une évaluation croisée. Vous avez dans ce but une grille d'évaluation qui servira de barême. Vous réaliserez l'évaluation la plus juste possible, la qualité de votre évaluation sera aussi l'objet d'une note.

Ce sera aussi l'opportunité pour vous d'identifier des pratiques pertinentes ou pas dans le travail de vos camarades.

3.2 Notice en anglais

Vous devrez aussi remettre une notice d'utilisation de votre programme en anglais, expliquant simplement et de façon concise (1 ou 2 pages) comment lancer le programme et utiliser l'interface de jeu. Cette notice sera évaluée par M. Delhoumi. Un seul rendu par binôme est demandé, vous penserez bien à vous enregistrer dans un binôme (même seul) afin d'accéder à l'interface de rendu.

4 Deuxième phase : moteur de jeu

Dans une deuxième phase, vous aurez à implémenter le moteur du jeu, en remplacement du moteur existant.

4.1 Sprint

Ce travail sera à effectuer au cours du deuxième sprint (en deux jours). Ce moteur devra respecter *scrupuleusement* les instructions et spécifications indiquées dans la documentation (`board.h`).

Ces interfaces seront déposées sous la forme d'un unique fichier `board.c` dans un module dédié qui testera votre code à l'aide de tests unitaires. Ces tests évalueront l'ensemble des fonctionalités indiquées dans la documentation. Votre code devra donc compiler parfaitement avec le `board.h` fourni (sans modification de votre part). Bien sûr, il est possible (et même recommandé) d'utiliser des fonctions intermédiaires dans votre code, ce qui ne sera pas visible de l'extérieur mais peut vous aider à faire un code plus facile à lire et ayant moins de bugs.

Vous pourrez travailler en binôme mais selon les mêmes modalités (et avec la même personne) que le sprint précédent).

Le barême sur les règles du jeu hors variante sera élevé, et il est donc plus important d'avoir une version où le jeu sans les variantes est implémenté très précisément que de faire des ébauches sur un jeu de base non fonctionnel.

Si vous ne souhaitez pas faire une implémentation d'une fonction, faîtes simplement une fonction vide (ou qui retourne 0) pour que les tests puissent s'exécuter.

4.2 Avec tests unitaires

Après avoir soumis votre code à l'issue du sprint, vous aurez la possibilité de ré-évaluer votre code autant de fois que vous le souhaitez avec un retour fourni sur les tests échouant. Ce retour vous permettra de retravailler votre code et de corriger les éventuels bugs, et l'évaluation de ce deuxième rendu sera elle aussi prise en compte dans la note.

Ce travail devra être déposé individuellement sur Caséine. Sans rendu individuel, le premier rendu sera évalué comme deuxième rendu aussi.

Bonne programmation!