Sujet

Contexte

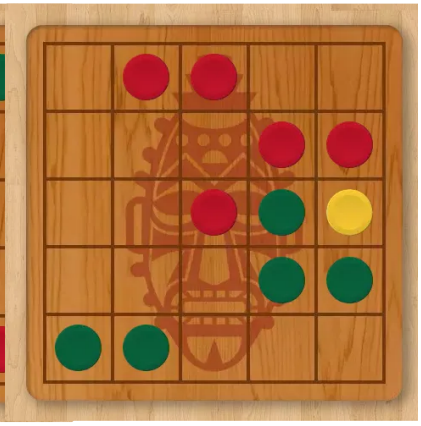
Le plateau de Bobail
(Crédit: https://boardgamegeek.com)

Le Bobail est un jeu de société africain. Il se joue sur un plateau de 5x5 cases. Chaque joueur dispose de 5 pions. Un dernier pion est neutre, le BOBAIL, et est installé au centre du plateau (jaune dans l'exemple ci-dessus).

Conditions de victoire

Pour gagner il faut amener le BOBAIL dans une position particulière:

Soit le bobail est positionné sur une des 5 cases de départ des pions. (Ici, c'est donc les verts qui gagnent)

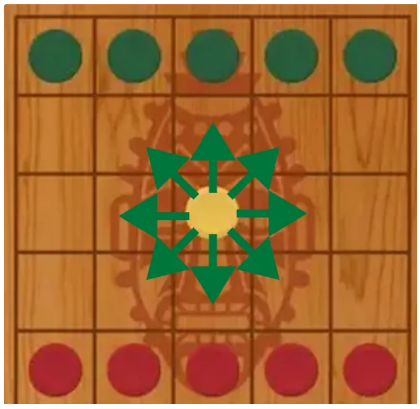


Soit en bloquant le BOBAIL (de sorte que l'adversaire ne puisse pas le bouger).

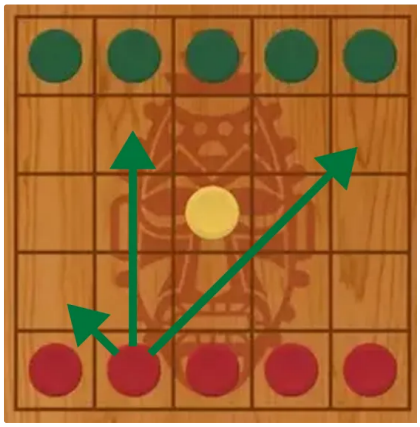
Règles de déplacement

Les règles de déplacement des pions sont les suivantes:

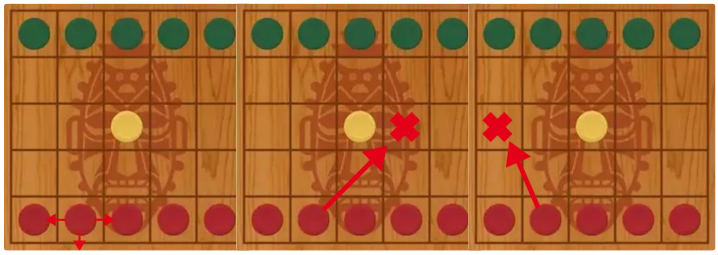
Le BOBAIL ne se déplace que d'une seule case, dans n'importe quelle direction (y compris les diagonales).



Les pions se déplacent également dans toutes les directions, mais jusqu'au bout de la rangée, ou jusqu'à un obstacle.



Sont donc illégaux :



Les mouvements de pion qui ne déplacent pas ce dernier d'au moins 1 case.

Les mouvements de pion (hors BOBAIL) qui ne vont pas jusqu'au bout de la rangée/ jusqu'à un obstacle.

Les mouvements de pion qui ne sont pas sur les diagonale, l'horizontale ou la verticale (évidemment).

Déroulé de la partie

Les joueurs jouent à tour de rôle. À son tour, le joueur déplace deux pions. D'abord le BOBAIL, puis un de ses 5 pions, suivant le principe de déplacement du jeu.

Exception : celui qui entame la partie n'avance que l'un de ses pions (le mouvement du BOBAIL est sauté).

Le jeu est disponible ici, si vous voulez tester votre compréhension des règles :

https://boardgamearena.com/gamepanel?game=bobail

Ps: Il faut créer un compte par joueur. Le site ne permet pas de jouer à deux derrière un seul écran…

Projet

Implémenter le moteur du jeu

Choisir une structure de données pertinente pour représenter l'état du jeu.

Implémenter l'interface par ligne de commandes.

Affichage de l'état du jeu avant chaque mouvement de pion et à la fin de la partie. Vous êtes libre de l'affichage (pensez à une interface facilement compréhensible pour 2 joueurs).

Demander le mouvement désiré par le joueur pour le BOBAIL (sauf au

tout premier coup de la partie) et pour un de ses pions. Libre à vous de choisir comment entrer les mouvements (example: de manière relative avec TOP(T), DOWN(D)… ou de manière absolue avec A1, B3… ou autre).

Effectuer le mouvement en mettant à jour la structure de données.

Vérifier que les coups sont légaux.

Vérifier les conditions de victoire (et afficher un message de fin de partie).

Implémenter une fonctionnalité avancée

Ajouter une “IA” que vous pourrez affronter

Implémenter une (ou plusieurs ?) méthode.s d’IA adaptée.s pour ce jeu (exemple: Monte Carlo Tree Search).

Ajouter une option à l’exécutable en ligne de commandes pour pouvoir jouer contre cette IA.

Ajouter une interface graphique

C’est quand même plus ergonomique pour jouer ^^.

Vous êtes totalement libre dans cette partie. Laisser libre cours à votre imagination. Faites vos propres recherches sur quel outils utiliser pour faire une GUI (Graphical User Interface). C’est aussi ça la programmation.

Notez que je ne recommande pas cette option pour ceux travaillant sur Windows. Cela rend l’installation des librairies graphiques plus complexe…

Autres

Si vous avez d’autres idées, n’hésitez pas ! (e.g. pouvoir communiquer par réseau avec un autre ordinateur pour jouer à distance…)

Critères d’évaluation et consignes

Vous devrez envoyer par mail **un lien Github** (accessible qu’en privée et en invitant les chargés de TD et le professeur) contenant votre projet complet (code source + compte rendu en pdf). Le mail devra avoir comme objet « Projet IN104 ». Vous pouvez utiliser le mail suivant:

guillaume1.thomas@protonmail.com

Veuillez nous inviter sur GitHub avec les pseudos suivants:

Whojo

nguyensaomai

Veuillez inclure un README sur le github qui explique comment compiler et exécuter votre projet sur un Ubuntu. Pour tester que cette étape fonctionne correctement, vous êtes encouragés à utiliser le serveur Ubuntu fourni par l’école ou à défaut une machine virtuelle comme VirtualBox (c’est simple et il y a plein de tutoriels en ligne). **S’il n’est pas possible de compiler et d’exécuter le projet dans un temps raisonnable (<5min) et sans erreur, la notation en pâtira.**

Faites donc attention au README, c’est important pour tout projet (y compris dans votre avenir professionnel). Sans cela, personne ne pourra réutiliser, ni ne s’intéressera à vos projets. Par ailleurs, vous pourrez ouvrir vos projets en public, une fois le projet terminé. Cela vous fera une belle vitrine pour vos stages.

Le compte rendu devra expliquer les différents choix de design que vous avez pris. Voici quelques idées:

vous avez pris. Voici quelques idées.

Comment avez-vous structuré votre code ?

Quelle.s structure.s de données ? Que contiennent-elles ? Quels avantages comparées à des alternatives ? (par exemple pour vérifier la condition de victoire ou pour effectuer un mouvement …)

Quelle arborescence de fichier ?

Détaillez les algorithmes utilisés pour vérifier la légalité d'un coup et les conditions de victoire.

Détaillez l'algorithme d' "IA" que vous avez utilisé. Quels hyperparamètres ? Quelles hypothèses ? Pourquoi celui-ci ?

Précisez aussi ce qui vous a posé des difficultés lors de ce projet et comment vous avez résolu ces derniers. Un bon programmeur/ ingénieur se distingue par la manière de gèrer ce genre de difficultés. Personne n'y arrive du premier coup. Je veux essayer de voir les raisonnements qui vous ont mené à la version actuelle de votre projet.

En plus des fonctionnalités du programme, la notation prendra en compte la lisibilité, la modularité et la maintenabilité de votre code. Voici quelques conseils pour l'améliorer:

Découper le code en sous fonction et en plusieurs fichiers de manière logique.

Faites attention à vos noms de variables/ fonctions, ils doivent être explicites.

Porter attention à ce que votre code se lise facilement et soit bien espacé.

Vous êtes encouragés à tester votre code (avec des tests unitaires automatisés et non simplement par ligne de commande).

Indentez votre code.

Utiliser des noms de commits explicites.

Utiliser des outils (Git, Framework de test, Code formatter, Code coverage…).

Toute démonstration de curiosité sur les notions de Clean Code ou d'outils sera valorisée. Amusez-vous !

Tips pour le projet

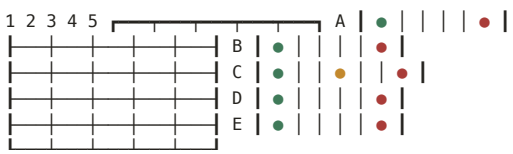
Représentation du plateau: Utilisez une `enum` pour représenter les pions plutôt que des entiers.

```
typedef enum { EMPTY, PLAYER_1, PLAYER_2, BOBAIL, }
Piece;
```

Arborescence de fichiers recommandée pour le projet:

```
bobail_project/ ├── src/ # Code source .c ├── include/ #
Fichiers d'en-tête .h ├── report/ # Rapport en pdf/
latex └── Makefile # Configuration de compilation
```

Utilisez les caractères Unicode "Box Drawing" pour un rendu graphique élégant dans le terminal. Exemple:



Comment commencer?

Créez le github et inviter votre camarades & les professeurs

Posez vous les questions suivantes. Faire un schéma grossier sur papier de la structure du code aide beaucoup.

- Quelles fonctions devrez-vous implémenter ?
- Quels arguments ?
- Que retournent-elles ?
- De quelles fonctions sont-elles dépendentes ?
- Comment les répartir dans différents fichiers ?
- Par lesquelles commencer ?
- Comment les tester ?

Ressources pour ceux qui souhaitent apprendre à mieux programmer

Example d'un bon README (avec présentation du projet, exemple avec des screenshots, comment l'installer et l'exécuter, noms des auteurs etc): https://github.com/matiassingers/awesome-readmeprovide the link to your repository in the body of the email.

Référence précise sur les fonctions de la librairie standard du C: Man Page

Cycle de développement avec Git

Why you should use Git (court article de blog en anglais)

Commit atomique (court article de blog en anglais)

Nomenclature pour les Commits

Makefile pour plus facilement compiler/ tester des projets avec une hiérarchie de fichiers

Philosophie du code propre

Introduction au Clean Code (Vidéo YouTube de 10min en anglais ⚠️)

Framework de tests: Criterion

Test Driven Development

- Pourquoi utiliser le Test Driven Development ?
  - Focalise notre attention sur une particularité d'une fonction (plutôt que d'essayer de gérer tous les cas d'une fonction) ⇒ Coder plus rapidement
  - Aide à définir les interfaces (i.e. quels arguments et quoi retourner)
  - Filet de sécurité lorsqu'on modifie le code (détecte instantanément les erreurs)
- Resources
  - Introduction au Test Driven Development (Vidéo YouTube de 5min en anglais)
  - Blog court de Martin Fowler

Outil de Code Coverage (i.e. quelle proportion du code est effectivement testée): GCov

Outil de formattage automatique du code (i.e. formater le code de manière plus lisible): Clang-format et extension VSCode