# Algorithmique et langage C

Projet

## Introduction et spécifications du projet

L'objet de ce projet est de coder un jeu mettant en scène un affrontement stratégique entre deux groupes de combattants, dans lequel le joueur pourra contrôler un des camps contre un autre joueur ou une IA.

Ce projet est à développer à l'aide d'un groupe de 4 personnes. Il est conseillé de réfléchir à la répartition de la charge et d'utiliser des outils de mise en commun du travail (Git, communication de groupe). En revanche, les outils de génération de code (CoPilot, ChatGPT, etc.) ne sont pas authorisés.
Le cadre de ce projet étant l'enseignement C, le seul langage autorisé est le C (pas de C++, notamment). Le programme devra pouvoir être compilé sur un ordinateur de démonstration fourni par le groupe lors de la soutenance, à l'aide de gcc ou d'un makefile approprié.

En plus du code qui est à rendre, il sera demandé au groupe une rapide présentation du programme qui fera intervenir chaque membre (supports éventuels laissés au choix du groupe).

Il est recommandé de lire le sujet entièrement avant de commencer le développement afin d'anticiper d'éventuelles contraintes de code.

# Description fonctionnelle

## Partie 1 - Le moteur de jeu

L'aire de jeu sera modélisée par une grille de 7 cases par 5. Sur ce plateau, 4 unités 'héros' sont placées d'un côté, et 4 autres 'ennemis' de l'autre (voir situation initiale d'exemple).

La boucle principale de jeu se déroule comme ceci :

- Le joueur courant sélectionne une de ses unités en indiquant des coordonnées de la grille.
- En sélectionnant de nouvelles coordonnées, l'unité est déplacée
- Si l'unité est assez proche, elle peut alors attaquer un adversaire.
- Une fois le mouvement (et l'attaque) terminés, l'unité sélectionnée est alors fatiguée. Elle ne pourra pas être à nouveau sélectionnée tant qu'il restera au moins une autre unité non-fatiguée dans le camp du joueur.
- La main passe alors au joueur suivant qui peut à son tour déplacer une unité.

Lorsque la dernière unité non-fatiguée d'un camp est jouée, tout le monde redevient à nouveau en forme. Cela implique qu'un camp qui ne dispose que d'une seule unité restante dispose d'un avantage : il pourra la jouer successivement y compris contre un adversaire ayant encore de nombreuses unités (qui est donc forcé d'attendre d'avoir fait jouer toutes ses unités une fois pour les utiliser à nouveau).
Lors d'une attaque, le pion attaquant et défenseur font une passe d'arme (l'attaquant inflige des dégâts, et le défenseur, s'il est encore en vie, contre-attaque). Les dégâts infligés sont égaux à la puissance du pion qui frappe réduits de la défense du pion qui est frappé (minimum 1).

Pour diversifier les unités mises dans l'aire de jeu, chaque camp possède 2 soldats (mis le plus en avant vers l'ennemi, B D W et Y dans l'exemple ci-dessous), un archer (mis en arrière et vers le haut, A et X dans l'exemple) et un filou (mis en arrière et vers le bas, C et Z dans l'exemple). Leurs caractéristiques sont en annexe.

La partie s'arrête lorsqu'un camp ne possède plus aucune unité sur le plateau, le camp restant étant alors vainqueur. Si après 100 coups, aucun joueur n'est gagnant, le camp avec le plus de points de vie gagne, puis le joueur qui n'a pas commencé en cas de nouvelle égalité.

**Exemples :**

Situation initiale

```
   1 2 3 4 5 6 7
  |-------------|
a | | | | | | | |
  |-------------|
b |A|B| | | |W|X|
  |-------------|
c | | | | | | | |
  |-------------|
d |C|D| | | |Y|Z|
  |-------------|
e | | | | | | | |
  |-------------|
```

Camp 1 : A, B, C et D
Camp 2 : W, X, Y et Z

Un mouvement

```
  1 2 3 4 5 6 7           1 2 3 4 5 6 7
 |-------------|         |-------------|
a| | | | | | | |        a| | | | | | | |
 |-------------|         |-------------|
b|A|B| | | |W|X|        b|A| | |B| |W|X|
 |-------------|         |-------------|
c| | | | | | | |   ->   c| | | | | | | |
 |-------------|         |-------------|
d|C|D| | | |Y|Z|        d|C|D| | | |Y|Z|
 |-------------|         |-------------|
e| | | | | | | |        e| | | | | | | |
 |-------------|         |-------------|
```

Sélectionner une unité : b2
Unitée sélectionnée : B. Indiquer la case à atteindre (max 2 cases) : b4
Indiquer une case où frapper (vide pour ne rien faire) :
B : b2 -> b4

Mouvement et attaque

```
  1 2 3 4 5 6 7           1 2 3 4 5 6 7
 |-------------|         |-------------|
a| | | | | | | |       a| | | | | | | |
 |-------------|         |-------------|
b|A| | |B| |W|X|       b|A| | |B|W| |X|
 |-------------|         |-------------|
c| | | | | | | |  ->   c| | | | | | | |
 |-------------|         |-------------|
d|C|D| | | |Y|Z|       d|C|D| | | |Y|Z|
 |-------------|         |-------------|
e| | | | | | | |       e| | | | | | | |
 |-------------|         |-------------|
```

Sélectionner une unité : b6
Unitée sélectionnée : w. Indiquer la case à atteindre (max 2 cases) : b5
Indiquer une case où frapper (vide pour ne rien faire) : b4
W : b6 -> b5, attaque B.
W inflige 2-1 = 1 dégâts à B.
B passe de 5 à 4 points de vie.
B contre-attaque
B inflige 2-1 = 1 dégâts à W.
W passe de 5 à 4 points de vie.

(il est laissé au groupe le loisir de modifier la présentation suggérée afin de la rendre plus ergonomique)

## Partie 2 - Fonctionnalités additionnelles

Afin de permettre une utilisation confortable du programme, un menu sera également ajouté avec au moins les fonctionnalités suivantes :

- Début d'une partie 1 joueur
- Début d'une partie 2 joueurs
- Début d'un test de deux IA
- Options
- > Taille de la grille
- > Choix de l'IA (prévision sur 1 ou 3 tours)
- A propos
- Quitter

Lancer une partie à deux joueurs permet à deux utilisateurs de contrôler tour à tour les deux camps.
Lancer une partie un joueur permet à ce joueur de se confronter à une intelligence artificielle (voir partie 4).
Lancer un test d'IA permet de comparer deux IA (voir partie 4b).
Le choix 'A propos' affichera la liste des développeurs du jeu ainsi que les bibliothèques libres de droit éventuellement utilisées (ainsi que toute autre information qui vous semble pertinente).

## Partie 3 - Aspect graphique

Afin de permettre une utilisation plus simple du programme, une interface graphique doit être mise au point. Pour ce faire, la bibliothèque RayLib devra nécessairement être utilisée.
Puisque l'aspect graphique d'un programme est avant tout une question subjective, le sujet se limitera aux exigences mesurables suivantes :

- Le programme doit être entièrement utilisable sans passer par la console d'exécution
- Le programme doit être utilisable avec le clavier et/ou la souris
- Le programme doit faire usage de texte, d'au moins 3 images différentes, et de couleurs.
- Le programme doit être raisonnablement réactif aux commandes utilisateur : pas plus de .5s de latence entre action et retour programme
- Le programme ne dois pas créer d'erreurs dans l'affichage des informations (eg. afficher une valeur erronée de points de vie, continuer à afficher une unité retirée du plateau...)

Enfin, afin de limiter les contraintes ne découlant pas de compétences de développement logiciel, l'usage d'images trouvées sur internet ou génerées pour le projet est autorisé.

## Partie 4 - Intelligence artificielle

Afin de permettre à un joueur seul de jouer, une intelligence artificielle sera mise au point. Une intelligence dite "minMax" est adaptée aux contraintes du jeu, et sera donc la piste poursuivie par le sujet.

Celle-ci fonctionne par notation de plateau : on suppose un coup (un mouvement, et une attaque si possible), et on donne une note à l'état de la partie en fonction du résultat. Une note élevée correspond à une situation favorable pour l'IA.
Après avoir étudié tous les coups possibles, on peut alors sélectionner le coup qui donne la meilleure note : le meilleur coup à jouer (en cas d'égalité de maxima, on prendra au hasard).

Cette fonction de notation devra prendre en considération ces critères :

- nombre de pions dans mon camp encore en vie : +1000 par pion
- points de vie totaux dans mon camp : +20 par point de vie
- nombre de pions dans le camp ennemi encore en vie : -500 par pion
- points de vie totaux chez l'ennemi : -10 par point de vie
- proximité des pions dans mon camp entre eux : -1 par case de distance
- proximité des pions dans mon camp avec l'ennemi : +0 par case de distance

Le meilleur coup possible ne prend pas toujours une stratégie à long terme en compte. Afin de minimiser ce problème, on va effectuer une prévision sur plusieurs coups.
On modélise le futur comme un arbre dont la racine est le présent. Chaque fils de cette racine est un des coups possibles, et comporte deux informations : le coup qui doit être joué pour arriver à ce noeud, et la valeur du tableau.
A partir d'un de ces noeuds, les feuilles qui en partent sont tous les coups possible par l'adversaire, ainsi que la notation du plateau, et ainsi de suite.
Ainsi, chaque étage de l'arbre corresponds à tous les futurs possibles après 1, 2, 3.. coups.

L'arbre complet ne peut pas être facilement construit, il faudra donc le développer au fur et à mesure de son parcours. A l'aide d'un parcours en largeur, on peut visiter tous les noeuds correspondant à un nombre de coups joués sans avoir besoin de construire l'arbre entier. A partir de ces données, si on suppose que l'adversaire fera le meilleur coup pour lui (donc le pire pour l'IA), on peut alors prévoir une stratégie sur plusieurs coups.

Afin de limiter les temps de calculs, on prendra une prévision sur 3 coups (un coup de l'IA, un coup de son adversaire, et un nouveau coup de l'IA). La restriction de .5s de latence ne s'appli-

quera pas aux prévisions IA sur plusieurs coups.

## Partie 4b - Algorithmes évolutifs

Afin de finaliser l'IA de notre jeu, on souhaite la voir se perfectionner. Afin de pouvoir faire cela, on va commencer par lui donner une existence qui persiste entre plusieurs parties : un fichier. Ce fichier contiendra la liste des points attribués aux différents critères de la partie 3.

L'évolution de l'IA à pour but de la rendre plus performante. Le critère que nous allons sélectionner sera de faire s'affronter deux IA. Celle qui sera le plus souvent vainqueur après 101 matchs est la plus performante.
A présent, faisons muter notre IA.
Pour ce faire, nous pouvons dupliquer son fichier de personnalité en altérant deux valeurs de notation de plateau entre -10% et +10% (minimum 1). Un affrontement entre cette nouvelle série de valeurs et l'ancienne déterminera le fichier à garder.

Après avoir répété ce processus un certain nombre de fois, que constatez-vous ? Que donne un test sur 101 matchs entre la version la plus aboutie de votre IA et la table de valeurs originale ?

# Annexes

## Grille de notation

- Code : 16 points
  - Compilation sans erreurs (1 point) ni warnings (1 point). Code commenté (1 point).
  - Partie 1 & 2
    - Sélection et déplacement des unités (2 points)
    - Implémentation des différentes unités (2 points)
  - Partie 3
    - Affichage du jeu (2 points)
    - Support clavier/souris (1 point)
    - Absence de bugs graphiques (1 point)
  - Partie 4
    - Possibilité de jouer contre une IA (1 point)
    - Notation correcte d'un plateau (1 point)
    - Prévision sur 3 coups (1 points)
  - Partie 4b
    - Affrontement entre 2 IA (1 points)
    - Algorithme génétique de perfectionnement d'IA (1 point)
- Présentation : 4 points

## Caractéristiques

- Caractéristiques d'un pion soldat
  - Point de vie max : 5
  - Attaque : 2
  - Défense : 3
  - Déplacement : 2 (cases)
  - Portée : 1 (case)
- Caractéristiques d'un pion archer
  - Point de vie max : 3
  - Attaque : 3
  - Défense : 1
  - Déplacement : 2 (cases)
  - Portée : 3 (cases)
- Caractéristiques d'un pion filou

- Point de vie max : 5
- Attaque : 4
- Défense : 0
- Déplacement : 4 (cases)
- Portée : 1 (case)

(Le but étant d'avoir un pierre-feuille-ciseau soldat > filou > archer > soldat)