# Le projet en quelques lignes

- Vous devez coder une application ou un jeu, décrit ci-après.
- Vous êtes par groupe d'au plus 4 personnes. Cette limite est non négociable. Allez sur la page des groupes pour choisir votre groupe. Chaque groupe est encadré par une personne.
- Votre projet est découpé en lots et en tâches. La partie gestion de projet est aussi importante que la partie technique. Vous devez vous organiser correctement. Les lots et les tâches sont détaillés plus bas dans ce sujet.
- Les encadrants utilisent ce site pour noter vos projets. Allez sur la page correspondante pour en savoir plus.

---

# Jeu à programmer

## Oriieflamme

*L'ENSIIE vit dans la tourmente depuis la fermeture de la porte interdimensionnelle. L'école doit maintenant réparer les dégâts écologiques et sociaux causés par la guerre entre les ENSIIE.*

*FISE, FISA, FC et Personnels administratifs et enseignants ont été appelés pour gérer et optimiser les ressources de l'école. Mais les divers débats sur la manière de procéder ont mené à la scission en 2 factions. Chacune doit maintenant montrer au directeur de quoi elles sont capables et laquelle sera choisie pour terminer le travail...*

Vous devez coder un jeu de carte à 2 joueurs, chaque joueur ou joueuse représentant une faction. La partie se joue en plusieurs manches. A chaque manche, les factions posent à tour de rôle des cartes pour proposer au directeur des idées. Les cartes sont ensuite révélées une à une pour se confronter les unes aux autres.

## Les manches

Le jeu se joue en deux ou trois manches. Chaque manche est remportée par une des deux factions. La faction qui remporte deux manches gagne la partie. Chaque manche se joue en 2 phases.

### Première phase

#### La main et la boîte à idées

Chaque faction dispose au début d'une manche d'une *boîte à idées*, une pile de cartes mélangées. Elle y pioche les 8 premières cartes qu'elle met dans sa main et qu'elle garde secrètes. Il n'y a pas de système de pioche dans le jeu, ces 8 cartes seront les seules qu'elle aura au cours de la manche. Cependant, si elle n'est pas satisfaite de son tirage, elle peut, **une fois dans l'ensemble des manches du jeu**, remettre sa main dans la boîte à idées, la remélanger et repiocher 8 cartes.

#### Exposer ses idées sur le plateau

Au début de la première phase, une faction, au hasard, est désignée *Première faction*. À la deuxième phase, l'autre faction est désignée première faction. À la troisième phase s'il y en a une, la première faction est de nouveau désignée au hasard.

Le plateau de jeu est une grille 2D de taille infinie. À tour de rôle, en commençant par la première faction, chaque faction dépose une carte *face cachée* sur une case libre du plateau. La carte doit toucher horizontalement ou verticalement une carte précédemment posée. Ainsi, à l'issue de cette phase, 16 cartes ont été posées sur le plateau et forment un ensemble connecté de cartes. Ces cartes sont posées faces cachées de sorte qu'aucune faction ne sait quelles cartes ont été posées par l'autre.

### Deuxième phase

On retourne les cartes une par une. On commence avec les cartes de la ligne la plus en haut, en retournant les cartes de gauche à droite sur cette ligne. Puis les cartes de la ligne suivante, toujours de gauche à droite, et ainsi de suite jusqu'à ce que toutes les cartes soient retournées. Chaque carte dispose d'un pouvoir qui est activé au moment où elle est retournée. Si le pouvoir ne peut être appliqué, alors rien ne se passe.

Certains pouvoirs permettent de supprimer des cartes du plateau ou de les retourner sans appliquer leur effet. On ne retourne donc plus ces cartes dans la suite du jeu. Certains pouvoirs ont pour effet de remettre des cartes face cachée en haut du plateau. Dans ce cas, au tour suivant, au lieu de retourner la carte qui aurait dû être retournée en temps normal, on commence par retourner ces cartes. De manière général, quand on doit choisir la prochaine carte à retourner, on choisi toujours la carte non retournée qui est sur la ligne la plus en haut et le plus à gauche possible sur cette ligne. Pour simplifier, cette carte sera nommée *carte la plus en haut à gauche du plateau*.

**Les points DDRS**

Certaines cartes attribuent des points DDRS à une faction ou en retire à la faction adverse. Il n'y a pas de limite au nombre de points que peut avoir une faction dans une manche mais elle ne peut en avoir moins de 0. Au début de chaque manche, les deux factions ont 0 points DDRS.

**Victoire de manche**

La faction qui remporte la manche est celle qui a le plus de points DDRS à l'issue de la manche. En cas d'égalité, la faction qui a posé la carte qui est la plus en haut à gauche du plateau à l'issue de la manche gagne *(attention, puisque certaines cartes peuvent être supprimées ou déplacées, la carte la plus en haut à gauche à la fin de la manche n'est pas nécessairement la carte qui a été retournée en premier durant la manche)*. On remélange ensuite les cartes de chaque faction dans sa boîte à idée avant de démarrer la manche suivante.

# Les cartes

Les deux factions disposent chacun d'une copie des mêmes cartes dans leur boîte à idée. Chaque carte dispose d'un pouvoir spécial qui est activé au moment où la carte est retournée sur le plateau. Enfin, chaque carte peut apparaître plusieurs fois dans la boîte et donc être piochée plusieurs fois.

| Nom de la carte | Effet de la carte | Nombre d'occurences |
|---|---|---|
| **FISE** | La faction qui a posé cette carte gagne 1 point DDRS. | 4 |
| **FISA** | La faction qui a posé cette carte gagne 2 points DDRS si le nombre de cartes retournées sur le plateau (y compris celle-ci) est pair, et 0 sinon. | 4 |
| **FC** | La faction qui a posé cette carte gagne 4 points DDRS si au moins une autre carte FC est retournée sur le plateau et 0 sinon | 4 |
| **EcologIIE** | La faction qui a posé cette carte gagne 1 point DDRS par carte FISE/FISA/FC retournée. | 2 |
| **lIIEns** | Prenez toutes les cartes FISE/FISA/FC retournées, retirez les du plateau, mélangez les et reposez les face cachées une par une sur la gauche de la carte la plus en haut à gauche du plateau, dans cet ordre. Les prochaines cartes à être retournées sont ces cartes là. | 2 |
| **Soirée sans alcool** | Si au moins une carte alcool est retournée, supprimez toutes les cartes FISE/FISA/FC retournées du plateau. Supprimez ensuite la première et la dernière ligne du plateau. Sinon la faction qui a posé cette carte gagne 5 points DDRS. | 1 |
| **Alcool** | Supprimez du plateau toutes les cartes qui touchent cette carte-ci (mais laissez la carte Alcool sur le plateau). | 1 |
| **Café** | Supprimez toutes les cartes Thé et Alcool retournées sur le plateau. Si une carte Ecocup est retournée sur le plateau, la faction qui a posé cette carte gagne 1 point DDRS. Sinon elle perd 1 point DDRS. | 3 |
| **Thé** | Supprimez toutes les cartes Café et Alcool retournées sur le plateau. Si une carte Ecocup est retournée sur le plateau, la faction qui a posé cette carte gagne 1 point DDRS. Sinon elle perd 1 point DDRS. | 3 |
| **Ecocup** | Cette carte est sans effet. | 1 |
| **Reprographie** | La faction adverse de celle qui a posé cette carte perd 1 points DDRS pour chaque paire de cartes retournées et identiques sur le plateau. (S'il y a 3 cartes identiques, cela fait 3 paires). | 1 |
| **Isolation du bâtiment** | Chaque faction gagne 1 point DDRS par carte non retournée et non supprimée du plateau qu'elle a posée sur le plateau. | 1 |
| **Parcours sobriété numérique** | Retournez toutes les cartes non retournées les plus à gauche et à droite de chaque ligne, sans appliquer leur effet. | 1 |
| **Heures supplémentaires** | La faction adverse de celle qui a posé cette carte perd 3 points DDRS par carte Heures supplémentaires retournée sur le plateau (y compris celle-ci). | 1 |
| **Kahina Bouchama** | Supprimez une carte non retournée du plateau choisie au hasard. | 1 |
| **Kevin Goilard** | Supprimez une ligne au hasard, la faction qui a posé cette carte gagne 2 points DDRS par carte supprimée ainsi. | 1 |
| **Massinissa Merabet** | La faction qui a posé cette carte réactive l'effet de la dernière carte retournée avant Massinissa Merabet, en faisant comme elle l'avait posée elle-même, même si ce n'est pas le cas. | 1 |
| **Vitéra Y** | La faction qui a le moins de points DDRS gagne 3 points DDRS. | 1 |
| **Jonas Senizergues** | Supprimez toutes les cartes Heures supplémentaires retournées du plateau. | 1 |
| **Fetia Bannour** | Si la carte Heures supplémentaires est retournée sur le plateau, supprimez toutes les cartes de la ligne et de la colonne où est posée cette carte (y compris celle-ci). Sinon la faction qui a posé cette carte gagne 1 point DDRS par carte Catherine Dubois, | 1 |

| | | |
|---|---|---|
| | Anne-Laure Ligozat, Guillaume Burel, Christophe Mouilleron, Thomas Lim, Julien Forest et Dimitri Watel retournée sur le plateau. | |
| **Catherine Dubois** | Supprimez la première et la dernière cartes de la ligne et de la colonne où est posée cette carte. | 1 |
| **Anne-Laure Ligozat** | Pour chaque carte EcologIIE, Ecocup, Isolation du bâtiment et parcours Sobriété numérique retournées, la faction qui a posé cette carte gagne 3 points DDRS et la dernière carte non retournée du plateau est supprimée. | 1 |
| **Guillaume Burel** | Si la faction adverse de celle qui a posé cette carte a plus de points DDRS, la seconde lui vole 3 points DDRS. | 1 |
| **Christophe Mouilleron** | Si la carte Heures supplémentaires est retournée sur le plateau, supprimez toutes les cartes retournées du plateau sauf les cartes Christophe Mouilleron et Heures supplémentaires. | 1 |
| **Thomas Lim** | Si Julien Forest n'est par retourné sur le plateau, la faction qui a posé cette carte gagne 3 points DDRS par carte FISE retournée sur le plateau. Sinon la faction adverse perd 1 point DDRS par carte FISE retournée sur le plateau. | 1 |
| **Julien Forest** | La faction qui a posé cette carte gagne 6 points DDRS par carte FISE retournée sur le plateau si au moins une carte Café est retournée sur le plateau. | 1 |
| **Dimitri Watel** | La faction qui a posé cette carte gagne 3 points DDRS par carte FISA ou FC retournée sur le plateau si au moins une carte Thé est retournée sur le plateau. | 1 |
| **Djibril-Aurélien Dembele-Cabot** | S'il y a plus de 3 cartes retournées sur la ligne de cette carte, la faction qui a posé cette carte gagne 5 points DDRS. | 1 |
| **Eric Lejeune** | Prenez au hasard 5 cartes retournées du plateau (ou toutes les cartes retournées du plateau s'il y a moins de 5). Si une de ces cartes est une carte Catherine Dubois, Anne-Laure Ligozat, Guillaume Burel, Christophe Mouilleron, Thomas Lim, Julien Forest ou Dimitri Watel, mélangez les et placez les à gauche de la case la plus à gauche de la première ligne. Les prochaines cartes à être retournées sont ces cartes là. Sinon, supprimez ces cartes du plateau. | 1 |
| **Lucienne Pacavé** | S'il y a une carte FISA retournée dans la même ligne ou la même colonne que cette carte, la faction qui a posé cette carte gagne 5 points DDRS. | 1 |
| **Katrin Salhab** | Si les cartes Djibril-Aurélien Djembele-Cabeau, Eric Lejeune et Lucienne Pacavé sont retournées, la faction qui a posé cette carte gagne 10 points DDRS. Sinon, retournez toutes les cartes dans la même ligne de cette carte sans appliquer leurs effets. | 1 |
| **Laurent Prével** | Si Laurent Prével est la dernière carte retournée du plateau, la faction qui a posé cette carte gagne la manche, quel que soit le nombre de points DDRS des deux factions. | 1 |

---

La suite décrit le travail à réaliser. Votre travail s'effectue par groupe d'au plus 4 personnes.

Le travail que vous devez rendre est découpé en 3 lots. Chaque lot est découpé en tâches, décrites ci-après avec chaque lot. Chaque tâche doit être attribuée à une seule et unique personne de votre groupe. Ces tâches doivent vous permettrent de vous répartir le travail en fonction de vos compétences et de vos envies. Chaque lot doit être rendu à une date précisée dans la page des évaluations. Toutes les tâches du lot sont ensuite évaluées par votre encadrant. La page des évaluations indique ce qui sera évalué, vous permettant ainsi de savoir ce qui est attendu.

Votre travail devra être développé et rendu avec l'outil de versionnement **git**. Cela vous permettra d'échanger facilement votre travail avec votre encadrant(e) et les autres membres de votre groupe. Cela permettra également à votre encadrant(e) de savoir qui a travaillé sur quelle partie du code.

Chaque personne de votre groupe devra travailler sur au moins 3 tâches (tous lots confondus). Ainsi, un groupe n'ayant que 2 membres devra rendre un travail correspondant à 6 tâches, moins conséquent qu'un groupe ayant 3 membres et devant rendre un travail correspondant à 9 tâches. Il est tout à fait possible, tant que vous respectez cette contrainte de ne pas rendre l'ensemble des lots décrits ci-dessous, mais plus votre travail est complet, meilleure sera votre note.

# Lot A

Le lot A consiste en la mise en place des interfaces. A l'issue de ce lot, vous n'aurez pas d'exécutable fonctionnel, juste une série de fichiers `.h`, un `Makefile` et le fichier `main.c` qui contiendra la fonction `main`. Vous ne devez coder aucune autre fonction que celle présente dans le fichier `main.c`. L'exécutable sera produit pour le lot B.

Le premier lot consiste à réfléchir à la mise en place du projet et à préparer les interfaces de vos modules (fichiers `.h`). Les interfaces ne contiennent que la définition de types (concrets ou abstraits), les signatures des fonctions utiles et des commentaires indiquant à quoi elles servent (mais pas comment elles seront codées). Elle ne contiennent pas de code à proprement parler. Pour les produire, il vous faut donc réfléchir, non pas à comment vous aller les implanter, mais ce que vous voulez que les fonctions fassent. À l’aide de ces signatures, vous pouvez programmer votre fichier principal `main.c` en le précompiler en fichier `main.o`. Ce fichier `main.c` doit contenir le code du logiciel permettant à un joueur de lancer une partie et de la jouer jusqu’à son terme selon le cahier des charges de la section précédente.

Vous ne serez pas en mesure de générer un exécutable à partir de `main.c`, puisque les fonctions décrites dans les

interfaces ne sont pas encore implantées. Mais ça ne vous empêche pas de les utiliser dans le fichier `main.c` puisque vous savez ce que ces fonctions sont sensées prendre en entrée et produire en sortie. Votre fichier devra compiler avec la commande `gcc -Wall -Wextra -std=c99 -c main.c`. Si vos interfaces et votre fichier `main.c` sont corrects, vous n'aurez plus qu'à implanter les fonctions des interfaces pour avoir un programme fonctionnel. Ces tâches sont prévues pour le lot B et le lot C. Cette méthode permet à votre chargé de s'assurer que vous ne partez pas dans une direction hasardeuse au début du projet. Le lot B constituera une implantation des fonctions de sorte à jouer en console. Le lot C constituera une évolution du projet (interface graphique, augmentation du nombre de fonctionnalités, ...).

Il s'agit donc essentiellement d'une phase de réflexion pour poser les bases de votre code. Il faut noter que l'architecture qui est proposée ici n'est pas la seule solution mais c'est celle que vous devrez respecter dans un premier temps même si elle ne vous semble pas la plus adaptée ou la plus utile. Voyez cela comme si votre client vous l'avait imposé. Sachez que plus vous avancerez dans le projet, plus vous aurez de libertés. Vous pourrez lors du lot C changer complètement cette architecture pour une autre qui vous convient mieux par exemple.

Attribuez chacune des tâches suivantes à une (et une seule) personne de votre groupe. Rappelez vous que chaque personne doit travailler sur au moins 3 tâches en tout.

# Tache A.1 - mise en place du projet et du main

Votre rôle est **central** pour le Lot A. Toute défaillance sur cette tâche peut affecter tout le travail du groupe. Il est important de la considérer **très** sérieusement.

Lisez les tâches de tous les membres du groupe avant de commencer.

- Mettez en place un nouveau dépôt git (n'utilisez pas le dépot du mini-TP tutoriel) et invitez votre encadrant sur votre dépôt en tant que rapporter ainsi que tous les membres du groupe en tant que developper. Ce git contiendra votre code. Créez y une première branche lot_a. Le
dernier commit de cette branche contiendra le code du Lot A une fois celui-ci terminé et déterminera la date à laquelle vous rendez votre lot.
- Créer un projet avec GanttProject et poussez le sur votre dépot git, dans la branche lot_a. Remplissez ce fichier avec les différentes tâches du Lot A. Mettez des durées arbitraires sur vos tâches. Ajoutez chaque membre du groupe comme ressource. Affectez ces membres aux tâches après décision de qui effectue quelle tâche. Enfin, indiquez les contraintes de précédences sur les tâches (quelle tâche doit être finie pour pouvoir commencer une nouvelle tâche). Vous pouvez couper les
tâches en sous-taches sur vous voulez.
- Codez le fichier `main.c`. Ce fichier devra utiliser toutes les fonctions et uniquement les fonctions qui auront été ajoutées dans les interfaces remplies par les autres membres du groupe. *Remarque : ce fichier restera normalement inchangé tout le long du Lot B*. Une fois ce dernier terminé, `main.c` pourra être compilé en un exécutable qui fera tourner le projet.
- Vérifiez, une fois toutes les interfaces rédigées par les autres membres, que votre fichier `main.c` compile sous forme de fichier objet `main.o`.
- Faites valider le Lot A par votre encadrant une fois toutes les tâches terminées.

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail. Vos commits doivent avoir l'indication Tâche A.1 (en plus d'un message décrivant le commit).

# Tache A.2 - carte.h et faction.h

Le fichier `carte.h` est en charge des informations des cartes du jeu : en particulier son nom, sa description, le nombre d'occurrences de cette carte. Il n'est pas responsable de la manipulation des cartes (placer une carte sur le plateau ou dans la main d'un joueur par exemple).

Le fichier `faction.h` est en charge des informations et des manipulations propres à une faction : son nom, le nombre de points DDRS, sa main, sa pioche ... mais pas ce qui affecte les deux factions à la fois. Par exemple ce n'est pas ce fichier qui décide quelle faction joue en premier dans une manche.

Ajoutez chacun des types et des fonctions suivants, avec les paramètres qui vous semblent adéquats à vous et au responsable de la tâche A.1 pour coder le fichier main.c. Documentez ensuite ces fonctions et ces types suffisamment pour pouvoir les implanter facilement dans le lot suivant et pour pouvoir les utiliser correctement dans le fichier `main.c`.

Dans le fichier `carte.h`:

- un type `carte` (possiblement abstrait)

Dans le fichier `faction.h`:

- un type `faction` (possiblement abstrait)
- une fonction permettant de savoir si une faction a utilisé l'option de remélanger sa main et la boîte à idée et de repiocher des cartes.
- une fonction permettant à une faction d'utiliser l'option ci-dessus.
- une fonction permettant à une faction de vider sa main dans sa pioche.
- une fonction permettant à une faction de mélanger sa pioche.
- une fonction permettant à une faction de repiocher ses cartes.

**Attention : aucune de ces fonctions n'échange d'informations avec le joueur humain (pas de printf, pas de**

**scanf). C’est le rôle de interface.h.**

On rappelle qu’un fichier .h ne contient pas de code, juste des signatures de fonctions.

Vous devez maintenir le dépot git à jour au fur et à mesure de l’avancement de votre travail. Vos commits doivent avoir l’indication Tâche A.2 (en plus d’un message décrivant le commit).

## Tache A.3 - plateau.h

Le fichier `plateau.h` est en charge de tout ce qui concerne le plateau et le déroulement de la partie : gérer les deux factions, poser les cartes sur le plateau faces cachées, les retourner et activer les effets. Il ne gère pas directement les cartes en main d'une faction, les scores des factions, les points de manche, ... qui sont gérés par le fichier `faction.h`. Il ne gère pas non plus les informations relatives aux cartes qui sont gérées par le fichier `carte.h`.

Ajoutez chacun des types et des fonctions suivants, avec les paramètres qui vous semblent adéquats à vous et au responsable de la tâche A.1 pour coder le fichier main.c. Documentez ensuite ces fonctions et ces types suffisamment pour pouvoir les implanter facilement dans le lot suivant et pour pouvoir les utiliser correctement dans le fichier `main.c`.

- un type `plateau` (possiblement abstrait)
- une fonction pour créer un nouveau plateau (et les deux factions qui joueront dessus)
- une fonction pour libérer la mémoire associée à un plateau (et ses deux factions)
- une fonction pour initialiser une nouvelle manche du jeu, ou, le cas échéant, indiquer que le jeu est terminé
- une fonction qui renvoie la liste des deux factions du jeu
- une fonction pour permettre à une faction de poser une carte face cachée sur le plateau
- une fonction pour retourner la carte la plus en haut à gauche face visible et activer son effet, et renvoyer cette carte

**Attention : aucune de ces fonctions n’échange d’informations avec le joueur humain (pas de printf, pas de scanf). C’est le rôle de interface.h.**

On rappelle qu’un fichier .h ne contient pas de code, juste des signatures de fonctions.

Vous devez maintenir le dépot git à jour au fur et à mesure de l’avancement de votre travail. Vos commits doivent avoir l’indication Tâche A.3 (en plus d’un message décrivant le commit).

## Tache A.4 - interface.h

Le fichier `interface.h` est en charge d'échanger des informations avec les joueurs. Par exemple demander quelle carte la faction veut poser sur le plateau et à quel endroit, ou afficher le plateau de jeu en console.

Ajoutez chacune des fonctions suivants, avec les paramètres qui vous semblent adéquats à vous et au responsable de la tâche A.1 pour coder le fichier main.c. Documentez ensuite ces fonctions et ces types suffisamment pour pouvoir les implanter facilement dans le lot suivant et pour pouvoir les utiliser correctement dans le fichier `main.c`.

Dans le fichier `interface.h`:

- une fonction pour afficher le plateau
- une fonction pour afficher la main d'une faction
- une fonction pour demander à une faction si elle veut utiliser son option permettant de vider sa main, de mélanger sa pioche et de repiocher
- une fonction pour demander quelle carte de sa main la faction souhaite poser face cachée sur le plateau
- une fonction pour demander à quelle position la faction souhaite poser sa carte
- une fonction pour afficher les effets d'une carte qui vient d'être retournée sur le plateau
- une fonction pour afficher le vainqueur de la partie

**Attention : aucune de ces fonctions ne modifie le plateau de jeu ou les factions ou ne contient d'information sur les cartes. C’est le rôle de plateau.h, faction.h ou carte.h.**

On rappelle qu’un fichier .h ne contient pas de code, juste des signatures de fonctions.

Vous devez maintenir le dépot git à jour au fur et à mesure de l’avancement de votre travail. Vos commits doivent avoir l’indication Tâche A.4 (en plus d’un message décrivant le commit).

# Lot B

Le but du Lot B est de permettre de compiler pleinement le fichier `main.c` en un exécutable fonctionnel. Il faudra ici implanter toutes les fonctions des interfaces créées dans le Lot A. Pour cela, un peu de travail est nécessaire pour permettre une bonne communication entre les différents modules du jeu. Par exemple, il est nécessaire d'accéder aux informations des cartes pour pouvoir les afficher, ce qui n'a pas été prévu dans le lot A puisque ce n'était pas nécessaire à la mise en place du fichier `main.c`. Ce lot s'intéresse également aux structures de données, à des fichiers de paramétrages (les constantes du jeu) et aux tests unitaires.

A l'issue de ce lot, vous pourrez normalement jouer à votre jeu.

Vous ne devez plus déplacer la branche `lot_a`. Elle restera définitivement pointée sur la fin de votre travail précédent même si vous remettez en question ce travail ultérieurement.

Attribuez chacune des tâches suivantes à une (et une seule) personne de votre groupe. Rappelez vous que chaque personne doit travailler sur au moins 3 tâches en tout.

# Tache B.1 - organisation du lot et makefile

Dans le dépot **git**, créez une deuxième branche `lot_b`. Le dernier commit de cette branche contiendra le code du Lot B une fois celui-ci terminé. Vous pouvez, vous ou les autres membres du groupe, si vous le souhaitez, créer d'autres branches temporaires le temps de la conception du Lot B.

Créer un fichier `makefile` et quatre dossiers `src`, `obj`, `bin` et `headers` pour ranger vos fichiers. Remplissez le `makefile` pour qu'il permette, à terme, de compiler chacun des fichiers sources en fichier objet (.o) et le fichier `main.o` en un exécutable.

Remplissez le fichier GanttProject avec les différentes tâches du Lot B. Mettez des durées arbitraires sur vos tâches. Affectez les ressources aux tâches après décision de qui effectue quelle tâche.

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail. Vos commits doivent avoir l'indication Tâche B.1 (en plus d'un message décrivant le commit).

# Tache B.2 - tests unitaires

Créez sur le dépot **git** une branche `lot_b_test` dans laquelle vous effectuerez vos tests sans affecter le code du `lot_b`. Créez un fichier `test.c` qui vous permettra de tester. N'hésitez par à modifier le code des autres et à commiter, il n'y a pas de risque, vous êtes sur une autre branche.

Vous pouvez utiliser CUnit si vous le souhaitez.

Effectuez les tests de l'exécutable. Remontez les bugs aux membres du groupe pour déterminer qui doit corriger ce bug. Vous pouvez par exemple commiter sur `lot_b_test` un code qui démontre le bug. Pensez à garder une trace écrite de vos tests, en particulier si le bug est visuel, s'il n'affecte pas le fonctionnement du jeu, s'il n'est pas codable, vous pouvez simplement décrire le bug dans un fichier texte que vous commiterez. Corrigez ou faites corriger ce bug sur la branche `lot_b`. Puis fusionnez les deux branches et recommencez.

Vos tests doivent inclure toutes les règles du jeu. Par exemple :

- Initialisation des factions
- Initialisation du plateau
- Ordre aléatoire des factions à la première manche
- Ordre déterministe des factions à la deuxième manche
- Présence ou non d'une troisième manche
- Option pour repiocher
- Poser une carte
- Placement des cartes sur l'espace 2D
- Appliquer les effets d'une carte (mise à jour des DDRS, impact sur le plateau, ..., inutile de - tester toutes les cartes une par une)
- Vainqueur de manche
- Vainqueur du jeu
- Démarrage du jeu
- Fin du jeu

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail. Vos commits doivent avoir l'indication Tâche B.2 (en plus d'un message décrivant le commit).

# Tache B.3 - structures de données

**Attention, on parle bien ici de structures de donnée (tableau, file, pile, ensemble, dictionnaire, ...), pas de structure au sens `struct` en C.**
Créez un nouveau fichier `structure.h` qui gérera toutes vos structures de données. Ces structures géreront le contenu du plateau, la liste des cartes en main, la liste des cartes dans la pioche et d'autres données organisées du jeu que vous jugerez utiles. Ces structures seront utilisées pour
implanter les types `carte`, `faction` et `plateau`.

Il peut s'agit de simples tableaux ou de structures plus évoluées. Ajoutez des types et des fonctions permettant de gérer ces structures (ajout, suppression, lecture du i-eme élément, ...). Documentez ensuite ces fonctions et implantez les dans un fichier structure.c.

Vous devez maintenir le dépôt git à jour au fur et à mesure de l'avancement de votre travail. Vos commits doivent avoir l'indication Tâche B.3 (en plus d'un message décrivant le commit).

# Tache B.4 - constantes et variables globales

Les règles du jeu disposent de beaucoup de constantes. Placez dans les fichier `carte.h`, `plateau.h` et `faction.h` ou d'autres fichiers adaptés des constantes ou des variables globales contenant les différents paramètres du jeu.

Par exemple le nombre de cartes dans la main d'une faction au début d'une manche, le nom des différentes cartes, le nombre de manches gagnantes, ... Ces constantes vous donneront plus facilement accès à ces informations et de manière plus lisible.

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail. Vos commits doivent avoir l'indication Tâche B.4 (en plus d'un message décrivant le commit).

## Tache B.5 - getters et setters

Implantez le type `carte` dans le fichier `carte.c`, le type `faction` dans le fichier `faction.c` et le type `plateau` dans le fichier `plateau.c`.

Ajouter aux fichiers `.h` associés des fonctions utiles pour récupérer des informations sur les cartes, les factions ou le plateau : par exemple récupérer le nom d'une carte, récupérer ou modifier le nombre de points DDRS d'une faction, savoir si la carte à la ligne 3 et colonne 7 du plateau est face visible ou face cachée, ...

Ces informations seront utilisées :

- par l'interface graphique
- par les autres fichiers lors du déroulement du jeu, en particulier lors de l'activation des effets d'une carte.

Implantez ensuite ces fonctions dans les fichiers `.c` associés.

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail. Vos commits doivent avoir l'indication Tâche B.5 (en plus d'un message décrivant le commit).

## Tache B.6 - carte.c, faction.c, plateau.c

Implantez toutes les fonctions restantes des fichiers `carte.h`, `faction.h` et `plateau.h` dans les `.c` associés, à l'exception des fonctions activant les effets d'une carte retournée qui seront codées dans la tâche B.7.

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail. Vos commits doivent avoir l'indication Tâche B.6 (en plus d'un message décrivant le commit).

## Tache B.7 - effets des cartes

Implantez toutes les fonctions permettant d'activer une carte retournée face visible sur le plateau.

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail. Vos commits doivent avoir l'indication Tâche B.7 (en plus d'un message décrivant le commit).

## Tache B.8 - interface

Implantez toutes les fonctions du fichier `interface.h` dans le fichier `interface.c`.

*Conseil : contentez vous dans un premier temps d'un affichage simple. Quand il fonctionnera, vous pourrez le faire évoluer vers une esthétique de meilleure qualité.*

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail. Vos commits doivent avoir l'indication Tâche B.8 (en plus d'un message décrivant le commit).

# Lot C

Le Lot C est complètement libre. Rédigez un petit rapport ou présentez oralement votre projet au chargé de projet, selon ce que vous et lui préférez.

Voici quelques idées :

- Vous pouvez tenter de mesurer l'impact environnemental de votre code. Par exemple en observant sa consommation mémoire avec top ou valgrind, le temps CPU utilisé avec time, ou les lectures écritures sur le disque avec l'outil iotop. Vous pouvez aller à fond et regarder la consommation électrique de la machine pendant l'exécution du jeu, mais il vous faut un wattmètre.
- Vous pouvez utiliser une bibliothèque d'interface graphique pour passer d'un jeu en console à un jeu graphique.
- Vous pouvez passer le jeu à un jeu en réseau
- Vous pouvez changer des règles. Par exemple créez des nouvelles cartes, ajouter des joueurs, ... Les changements doivent être drastiques. Une modification mineure ne sera pas une contribution très intéressante.

Organisez votre Lot en tâches, répartissez les tâches comme dans les autres lots.
Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail.

Attribuez chacune des tâches suivantes à une (et une seule) personne de votre groupe. Rappelez vous que chaque personne doit travailler sur au moins 3 tâches en tout.

## Tache C.1

Tache libre

## Tache C.2

Tache libre

## Tache C.3

Tache libre

## Tache C.4

Tache libre

---