Projet informatique 2019-2020

Attention ! Ce sujet n'a pas vocation à être imprimé. (D'ailleurs, il n'a probablement pas de vocation du tout puisqu'il y a peu de chance que ce sujet soit conscient.)

1 Jeux à programmer

1.1 Durableverse

On souhaite coder un jeux de type Trading Cards Game nommé Durableverse. Il s'agit d'un jeu de combat de cartes à collectionner.

À force de jouer avec le supercalculateur de l'école, les élèves du parcours CIDM ont ouvert une porte vers un univers parallèle, dévoilant ainsi l'existence d'une seconde école ENSIIE et de tout un tas d'autres écoles d'ingénieur. Avec effroi, les deux ENSIIE virent leur classement parmi les écoles d'ingénieur les plus investies dans le développement durable chuter, naturellement englouties sous un flot d'énergie démentiel pour maintenir la porte ouverte entre les mondes. C'est ainsi que débuta la bataille pour devenir l'école la plus investie de tout le multivers...

1.2 Les joueurs

Chaque joueur joue une ENSIIE. Une ENSIIE dispose :

— de points de développement durables (les *DD*)

— d'un deck de cartes, faces cachées, dans lequel elle pioche

— d'une main, où vont les cartes piochées

— d'un plateau de jeu où vont ses cartes

— d'*Energie* (les *PE*) pour poser des cartes de sa main vers le plateau de jeu.

— d'une défausse où sont posées les cartes qui ne sont plus en jeu.

1.3 Déroulement et conditions de victoires

Chaque ENSIIE dispose au départ de 0 points de développement durable (des *DD*) et cherche à monter ce nombre au delà de 20. Pour ce faire, elle dispose de cartes qu'elle pourra jouer pour monter son score ou diminuer celui de l'adversaire.

Le jeu se déroule en tours. Chaque tour est découpé en deux phases, une par ENSIIE. Au premier tour, une des ENSIIE (nommée A) effectue sa phase, pendant laquelle elle peut jouer diverses cartes ; puis la seconde école (nommé B) effectue sa phase. À la fin du tour, on compare l'état des ENSIIE A et B, et chacune gagne un certain nombre de DD (voir plus loin). Au tour suivant, B effectue sa phase avant A. Au tour suivant, A effectue sa phase avant B, et ainsi de suite jusqu'à la fin du jeu. L'ENSIIE commençant le jeu est déterminée au hasard.

Une ENSIIE l'emporte si, après la fin du tour, elle a plus de 20 DD et qu'elle a plus de DD que l'autre ENSIIE. Ainsi si l'une des deux a plus de 20 DD mais pas l'autre, alors la première gagne. Si les deux ont plus de 20 DD alors celle qui en a le plus l'emporte. Si les deux sont à égalité, un match nul est déclaré. Enfin, au bout de 30 tours, si aucune ENSIIE n'a gagné, la partie est annoncée comme nulle.

1.4 Les cartes, les decks et la pioche

Le jeu dispose de 3 types de cartes :

— les *Élèves*

— le *Personnel*

— les *Actions*

Il existe 2 cartes *Élèves*, une vingtaine de cartes *Personnels* et une vingtaine de cartes *Actions*.

Chaque ENSIIE dispose d'un deck contenant l'ensemble des cartes *Personnels* et *Actions* mélangées. Les cartes *Élèves* ne font pas partie du deck. **Au début du jeu**, chaque ENSIIE pioche 2 cartes. **Au début de sa phase du jeu**, une ENSIIE pioche une carte. Chaque ENSIIE commence donc sa première phase avec 3 cartes en main.

1.4.1 Cartes élèves

Il existe deux types de cartes *Élèves* : la carte *FISE* et la carte *FISA*. Ces cartes sont centrales. Elle permettent de gagner des DD et de gagner de l'énergie pour utiliser les cartes *Personnels* et *Actions*.

Poser une carte *Éleve* : Au début de sa phase, après avoir pioché, une ENSIIE a le choix entre mettre en jeu une carte *FISE* ou une carte *FISA*. Il n'y a aucun coût à payer. En posant ces cartes, le joueur doit former deux piles, une pile sur laquelle il pose les cartes *FISE* et une seconde pile sur laquelle il pose les cartes *FISA*.

Particularité des cartes FISA : Lors des tours pairs (le premier tour est le tour 1), les cartes *FISA* disparaissent du plateau et réapparaissent au tour (impair) suivant. Pendant les tours pairs, il est possible de choisir de mettre en jeu une carte *FISA*, mais cette carte disparait dès qu'elle est mise en jeu.

Points de développement et de durabilité : Une carte *Élève*, lorsqu'elle est posée, a *1 point de Développement* et *1 point de Durabilité*. **À la fin d'un tour** (lorsque les deux ENSIIE ont fini leur phase respective), chaque ENSIIE compte l'ensemble de ses points de Développement et soustrait à cette valeur l'ensemble des poids de Durabilité de son adversaire. Si ce nombre est positif, il gagne autant de DD. Attention, lors des tours impairs, les cartes *FISA* ne comptent pas. Par exemple, si ENSIIE A a 1 *FISE* et 3 *FISA* ; et ENSIIE B a 2 *FISE*, alors ENSIIE A gagne 2 DD lors d'un tour impair ; mais 0 lors d'un tour pair ; et ENSIIE B gagne 1 DD lors d'un tour pair et 0 lors d'un tour impair. Ces points de Développement et de Durabilité sont susceptibles d'être modifiés par les autres cartes du jeu.

Points d'énergie (PE) : Une carte *Élève* apporte de l'énergie pour jouer des cartes *Personnel* et *Action*. Après avoir posé sa carte *Élève*, le joueur dispose d'un PE par carte *FISE* et de deux PE par carte *FISA* lors des tours impairs. Lors des tours pairs, il ne dispose que d'un PE par carte *FISE*. Attention, les PE ne s'accumulent pas d'un tour à l'autre, il sont remis à zéro au début de chaque phase.

1.4.2 Les cartes *Personnels*

Les cartes *Personnels* permettent d'influer sur le jeu, par exemple en améliorant les caractéristiques de ses cartes *Élèves* ou en diminuant celles des cartes adverses. Il existe de nombreuses cartes variées.

Coût : Une carte *Personnel* possède un coût. Il faut payer autant de PE que le coût de la carte pour pouvoir la jouer. Si l'ENSIIE ne possède pas autant d'énergie, elle ne peut pas jouer cette carte.

Emplacements des cartes *Personnel* : Lorsqu'on joue une telle carte, après avoir payé son coût en énergie, l'ENSIIE peut la placer sur son plateau de jeu, à côté des deux piles de cartes *Élèves*. **Au début du jeu**, une ENSIIE ne peut avoir qu'une carte *Personnel* sur son plateau. **À partir du 6^{e} tour de jeu (inclu)**, chaque ENSIIE peut avoir, en même temps, 2 cartes de *Personnel*. **À partir du 11^{e} tour de jeu (inclu)**, chaque ENSIIE peut avoir, en même temps, 3 cartes de *Personnel* en jeu. Si l'ENSIIE joue une carte *Personnel* alors que tous les emplacements disponibles sont pleins, alors la carte *Personnel* qui a été jouée en premier est envoyée dans la défausse puis remplacée par la nouvelle carte.

Effets des cartes *Personnel* : Chaque carte *Personnel* a un effet tant qu'elle est en jeu. **Dès qu'elle quitte le jeu, son effet disparaît.** Le tableau suivant indique pour chaque carte *Personnel* son nom, son coût et son effet. Il existe dans le deck une carte de chaque type. Il existe 12 effets possibles, décrits après le tableau.

N°	Nom	Coût	AE1	AE2	AA1	AA2	RE1	RE2	RA1	RA2	ADD	RDD	DR	E
1	Thomas Lim	3	1											
2	Marie Szafranski	3		1										
3	Alain Faye	3			1									
4	Christophe Mouilleron	3				1								
5	Stefania Dumbrava	3					1							
6	Julien Forest	3						1						
7	Nicolas Brunel	3							1					
8	Laurence Bourard	3								1				
9	Dimitri Watel	5	1		1									
10	Vitera Y	5		1		1								
11	Kevin Goilard	5					1		1					
12	Vincent Jeannas	5						1		1				
13	Massinissa Merabet	7											2	
14	Anne-Laure Ligozat	8									2		1	1
15	Catherine Dubois	8										2	1	1
16	Eric Lejeune	10	1	1				1				1		
17	Christine Mathias	10			1	1				1		1		
18	Katrin Salhab	15	2	2										1
19	Abass Sagna	15			2	2								1
20	Laurent Prével	20	2	2	2	2					1	1	1	2

Les différents effets sont :

Code	Effet (où X est la valeur indiquée dans le tableau ci-dessus)
AE1	Ajouter X points de Développement à chacune de vos cartes *FISE*
AE2	Ajouter X points de Durabilité à chacune de vos cartes *FISE*
AA1	Ajouter X points de Développement à chacune de vos cartes *FISA*
AA2	Ajouter X points de Durabilité à chacune de vos cartes *FISA*
RE1	Retirer X points de Développement à chacune des cartes *FISE* de l'adversaire
RE2	Retirer X points de Durabilité à chacune des cartes *FISE* de l'adversaire
RA1	Retirer X points de Développement à chacune des cartes *FISA* de l'adversaire
RA2	Retirer X points de Durabilité à chacune des cartes *FISA* de l'adversaire
ADD	Gagner X DD de plus à la fin du tour
RDD	L'adversaire retranche X DD au total gagné à la fin du tour
DR	Piochez X cartes de plus au début de votre phase
E	Mettez en jeu X cartes *Etudiant* de plus au début de votre phase

FAQ : Une carte *Élève* ne peut avoir un nombre négatif de points de Développement ou de Durabilité. Si, dans la même phase, deux cartes *Personnel* font respectivement gagner et perdre des points à une carte *Élève*, on ajoute d'abord les gains, puis on retranche les pertes, et si le résultat est négatif, on le ramène à 0. Ainsi un élève qui gagnerait 2 points de Développement et en perdrait 4 aurait au final 0 points de développement.

FAQ : L'ENSIIE ne peut avoir moins de 0 DD. Si, dans le même tour, il gagne et perd des DD, alors, comme pour les cartes *Élèves*, on ajoute les gains, on retranche les pertes et si le résultat est négatif, on ramène à 0.

Disclaimer : Ce jeu se veut avant tout ludique et bon enfant. S'il est vrai (voire évident) que certains effets ont été associés à des membres du personnel à cause de leurs fonctions, une bonne partie a été positionnée arbitrairement. Le tableau ci-dessus ne veut, en aucun cas, signifier qu'une personne affecte plus positivement ou plus négativement le développement durable à l'ENSIIE et les étudiants de l'école.

1.4.3 Les cartes *Actions*

Les cartes *Actions* ont un effet immédiat et instantané sur le jeu. Une fois jouée, la carte va immédiatement dans la défausse.

Coût : Une carte *Action* possède un coût. Il faut payer autant de PE que le coût de la carte pour pouvoir la jouer. Si l'ENSIIE ne possède pas autant d'énergie, elle ne peut pas jouer cette carte.

Effets des cartes *Action* : Chaque carte *Action* a un effet qui est appliqué au moment où elle est jouée. Le tableau suivant indique pour chaque carte *Action* son nom, son coût, le nombre de fois qu'elle existe dans le deck, et son effet.

N°	Nom	Coût	Quantité	Effet
21	Cours Développement Durable	2	3	Gagnez 1 DD
22	Recrutement	2	3	Piochez une carte
23	Rentrée FISE	3	2	Mettez en jeu une carte *FISE*
24	Rentrée FISA	3	2	Mettez en jeu une carte *FISA*
25	Energie verte	4	2	Gagnez 6 PE.
26	Diplomation	5	2	Retirez une carte *FISE* et une carte *FISA* du plateau de l'adversaire
27	Décharge	5	4	Mettez la première des cartes *Personnel* mise en jeu par l'adversaire dans sa défausse.
28	Recyclage	10	1	Mélangez votre défausse et cette carte avec votre pioche
29	Zero papier	10	1	Vos cartes *Élèves* gagnent un point de Développement. Toute nouvelle carte *Élève* arrivant en jeu gagne 1 point de Développement.
30	Repas végétarien	10	1	Vos cartes *Élèves* gagnent un point de Durabilité. Toute nouvelle carte *Élève* arrivant en jeu gagne 1 point de Durabilité.
31	Fermeture annuelle	10	1	Retirez de tous les plateaux toutes les cartes *FISE* et *FISA*

1.5 Cahier des charges

Votre objectif est de coder le jeu Durableverse. Votre jeu devra permettre, à minima :

— De démarrer une partie et de la jouer jusqu'à son terme ;

— D'afficher l'état du jeu (cartes posées, cartes en main, points de développement durable, mana, joueur dont c'est le tour et divers messages du jeu) à chaque tour ;

2 Travail à réaliser.

Dans ce projet, il vous est demandé de respecter les consignes de chaque lot et d'utiliser les outils qui vous ont été présentés en cours. Le travail est découpé en 3 lots, le Lot A, le Lot B et le Lot C. Les deux premiers lots sont découpés en 12 tâches (4 pour le Lot A et 8 pour le Lot B). Vous êtes libres de découper le Lot C comme bon vous semble.

Chaque étudiant doit effectuer au moins **3 tâches**. Idéalement, un groupe de 4 élèves doit donc effectuer au moins les lots A et B, chacun 1 tâche du lot A et chacun 2 tâches du lot B. Une tâche vous sera attribuée et validée à 2 conditions : vous êtes désigné sur votre fichier GanttProject comme ressource de la tâche ; et vous avez commité la totalité du code relatif à cette tâche ou presque (une autre personne peut vous aider localement, corriger un bug ou une faute d'orthographe, ...). Un élève qui n'aurait pas effectué 3 tâches sans excuse valable se verra attribué la note de 0. Aucune tâche n'est sensée être trop longue. N'hésitez pas à demander de l'aide si vous en avez besoin.

Attention : on vous demande d'effectuer les tâches, pas d'être parfait, vous avez le droit à l'erreur.

Les outils Doxygen et CUnit sont facultatifs. Ils moins prioritaires que les autres. Toutefois :

— votre code doit être commenté !

— votre code doit compiler avec `gcc -Wall -Wextra -std=c99` sans erreur ni warning !

— votre code doit être testé !

Un code non commenté, un code qui ne compile pas, ou un code dont l'exécution plante au démarrage se verra attribué la note de 0.

2.1 Travail en équipe, référent du projet

Remarque importante : vous êtes 4 dans votre groupe. Répartissez vous les tâches selon vos compétences respectives. Cependant, il est important que chacun d'entre vous connaisse l'avancement du projet.

Pendant chaque séance, quand votre chargé de projet fait le tour des groupes, il peut demander à l'un d'entre vous d'être référent du groupe. Il s'agit alors de vous mettre dans la peau d'un manager technique qui présenterait son projet à une personne extérieure (un client, un supérieur, ...).

— Le référent doit pouvoir présenter où en est votre projet par rapport à la mise en place du lot en cours

— Il doit savoir qui fait quoi

— Il doit exposer quelques idées originales de l'équipe

— Il doit être capable de dire, sans rentrer dans les détails, si vous rencontrez des problèmes, qui en particulier et sur quel sujet.

— Cet exposé doit être court, il n'est pas nécessaire d'y passer plus de quelques minutes. C'est une synthèse, pas un rapport technique de 40 pages.

— Le référent est seul quand il parle, il ne peut pas recevoir de l'aide de ses associés.

Ceci implique donc une présence de chacun en salle de projet et une implication minimum de chacun dans le projet. Il n'est pas nécessaire que vous connaissiez tout le code du projet par coeur, vous devez juste être capable de synthétiser. Toute absence doit être justifiée. Pensez également, si possible, à prévenir votre groupe voire votre chargé de projet de votre absence, par politesse. Vous travaillez en équipe : si des problèmes surviennent dans votre groupe (absence prolongée d'un membre, tensions, ...), il vaut mieux en informer votre chargé de projet ou votre responsable d'UE pour remédier au problème rapidement ou trouver une solution alternative.

2.2 Lot A

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 leur fonction (et non leur implantation interne). 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. 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, ...).

Attribuez chacune des tâches suivantes, à une personne du groupe. Seule la tâche A.1 est attribuée au référent du projet. N'hésitez pas à découper chaque tâche en sous-tâches.

2.2.1 Tâche A.1 – Référent du projet – Mise en place du projet et `main.c`

Votre rôle est central pour le Lot A. Lisez les tâches de tous les membres du groupe avant de commencer.

Mettez en place le dépôt git 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 une première branche `lot_a`. Le dernier commit de cette branche contiendra le code du Lot A une fois celui-ci terminé.

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 les ressources aux tâches après décision de qui effectue quelle tâche.

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, que votre fichier `main.c` compile.
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.

2.2.2 Tâches A.2 et A.3 – Rédigez `plateau.h` et `carte.h`

L'interface `plateau.h` est en charge du plateau de jeu. Elle doit permettre d'accéder à l'ensemble des cartes présentes sur le plateau, la main, les decks et la défausse de chaque ENSIIE. Elle doit également permettre de modifier ce plateau en fonction des décisions prises par les ENSIIE au cours du jeu.

L'interface `carte.h` permettra de manipuler les cartes du jeu. Elle est très sommairement définie ici, seul le type `carte` y est défini.

Aucune de ces fonctions n'échange d'informations avec les joueurs humain (pas de `printf`, pas de `scanf`). C'est le rôle de `interface.h`.

Ajoutez chacun des types et des fonctions suivants, avec les paramètres qui vous semblent adéquats à vous et à votre référent pour coder le fichier `main.c` de la tâche `A.1`. Documentez ensuite ces fonctions.

Tâche A.2

- le type `carte` qui permettra de manipuler chaque carte du jeu (accéder au coût ou à l'effet d'une carte)
- le type `plateau` qui permettra de manipuler l'ensemble des cartes présentes sur le plateau de jeu (ceci inclu la main, le deck et la défausse de chaque ENSIIE). La plupart des fonctions qui suivent prendront un plateau en entrée et le modifieront ou renverront des informations relatives à ce plateau. Ce type peut être abstrait. Il sera alors défini concrètement dans le Lot B.
- une fonction pour créer un nouveau plateau
- une fonction pour libérer la mémoire attribuée à un plateau
- une fonction pour signifier au plateau qu'un nouveau tour commence. Elle permettra par exemple d'incrémenter le compteur de tour, de gérer les carte `FISA` et de mettre à jour les espaces disponibles pour les cartes `Personnels` sur le plateau.
- une fonction pour calculer combien de cartes seront piochées par une ENSIIE au début de sa phase.
- une fonction pour qu'une ENSIIE pioche une carte

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail.

Tâche A.3

- une fonction pour calculer combien de cartes `Elève` une ENSIIE recevra au début de sa phase.
- une ou deux fonctions pour ajouter une carte `Élève` de type `FISE` ou `FISA` au plateau de jeu d'une ENSIIE. *Attention, c'est le rôle de `interface.h` de demander au joueur s'il souhaite une carte `FISE` ou `FISA`.*
- une fonction pour calculer le nombre de PE disponibles par une ENSIIE après avoir posé sa ou ses nouvelles cartes `Élèves`.
- une fonction pour permettre à une ENSIIE de jouer une carte de sa main
- une fonction pour signifier au plateau que le tour est terminé. Elle permettra, entre autres, de faire le calcul des DD gagnés par chaque ENSIIE à la fin du tour.
- une fonction pour déterminer si la partie est finie, et si oui si un joueur a gagné ou s'il y a égalité.

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail.

2.2.3 Tâche A.4 – Rédigez `interface.h`

L'interface `interface.h` est en charge de tous les échanges entre les joueurs humains et le jeu : afficher le plateau, demander si le joueur veut une carte `FISE` ou `FISA`, demander s'il souhaite finir sa phase, ...

`Aucune de ces fonctions ne modifie le plateau de jeu. C'est le rôle de plateau.h.`

Ajoutez chacun des types et des fonctions suivants, avec les paramètres qui vous semblent adéquats à vous et à votre référent pour coder le fichier `main.c` de la tâche `A.1`. Documentez ensuite ces fonctions.

- une fonction pour afficher en console un message signifiant qu'un nouveau tour commence ; avec toutes les informations utiles (notamment si un nouvel espace pour les cartes `Personnel` est apparu)
- une fonction pour afficher en console un message signifiant qu'une nouvelle phase commence et l'ENSIIE associée à cette phase

— une fonction pour afficher le plateau

— une fonction pour demander à un joueur qui reçoit une nouvelle carte `Elève` s'il souhaite ajouter une carte `FISE` ou `FISA`.

— une fonction pour demander à un joueur de choisir une carte de sa main ou de finir sa phase. La fonction ne proposera que des cartes de la main qui coûtent moins cher que le nombre de PE du joueur. Si aucune carte n'est moins chère, alors seul le choix de finir sa phase est proposé au joueur.

— une fonction qui affiche le gagnant du jeu ou égalité le cas échéant.

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail.

2.3 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.

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 personne du groupe. Les tâches B.1 et B.8 seront attribuées aux référents du projets (deux personnes distinctes). N'hésitez pas à découper chaque tâche en sous-taches.

2.3.1 Tâche B.1 – Référent du projet – Lier tous les fichiers

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.

2.3.2 Tâche B.2 – Structures de données – `structures.h`

Créez un nouveau fichier `structure.h` qui gérera toutes vos structures de données.

Ajoutez des types et des fonctions permettant de gérer ces structures (ajout, suppression, lecture du `i-eme` élément, ...). Documentez ensuite ces fonctions.

Vous êtes libre d'utiliser les structures que vous voulez mais il vous est très fortement conseillé de coder une structure de `file` (first in last out) et éventuellement une structure de `pile` (first in first out) qui seront adaptées au projet.

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail.

2.3.3 Tâche B.3 – `structures.c`

Implantez toutes les fonctions du fichier `structures.h` dans un fichier `structures.c`.

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail.

2.3.4 Tâche B.4 – Getters de `carte.h` et `carte.c`

Implantez le type `carte` dans le fichier `carte.c`.

Ajoutez à `carte.h` des fonctions utiles pour récupérer des informations sur les cartes : son coût en PE, est-ce qu'il s'agit d'une carte `Personnel`, `Action`, ... ?, la quantité présente dans le deck au départ, ... En d'autres termes, toutes les fonctions qui vous semblent adéquates à vous et à votre référent pour coder le fichier `plateau.c` et le fichier `interface.c` des tâches `B.6` et `B.7`. Documentez ensuite ces fonctions.

Implantez ces fonctions dans `carte.c`.

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail.

2.3.5 Tâche B.5 – Getters de `plateau.h` et `plateau.c`

Implantez le type `plateau` dans le fichier `plateau.c`.

Ajoutez à `plateau.h` des fonctions utiles pour récupérer des informations sur le plateau : récupérer le deck d'une ENSIIE, récupérer la liste de ses cartes en main, sa défausse, le nombre de cartes FISE ou FISA, est-ce qu'on est sur un tour impair ou pair, ... En d'autres termes, toutes les fonctions qui vous semblent adéquates à vous et à votre référent pour coder les autres fonctions du fichier `plateau.c` et le fichier `interface.c` des tâches `B.6` et `B.7`. Documentez ensuite ces fonctions.

Implantez ces fonctions dans `plateau.c`.

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail.

2.3.6 Tâche B.6 – `plateau.c`

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

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail.

2.3.7 Tâche B.7 – `interface.c`

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

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail.

2.3.8 Tâche B.8 – Référent du projet – Tester votre code

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. Si le bug est visuel, s'il n'affecte pas le fonctionnement du jeu, s'il n'est pas codage, vous pouvez simplement décrire le bug oralement sans commiter.

Corrigez 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 :

- démarrage sans problème du jeu
- vérification des fuites mémoires avec valgrind
- ordre correct des phases des joueurs
- ajout d'une carte `FISE` ou `FISA`
- disparition des `FISA` un tour sur deux
- augmentation du nombre d'espace pour le `personnel` aux tours 6 et 11
- décompte des PE
- décompte des DD à la fin d'un tour
- effets des différentes cartes `personnels`
- effets des différentes cartes `actions`
- disparition de la première carte `personnel` jouée si tous les espaces disponibles sont pris et qu'une carte `personnel` est jouée,
- tout autre idée de votre part ...

Vous devez maintenir le dépot git à jour au fur et à mesure de l'avancement de votre travail.

2.4 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 réduire l'impact environnemental de votre code. Par exemple en observant sa consommation mémoire avec `top` ou `valgrind`, le temps CPU utilisé avec `time`, ou de diminuer 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. Peut être qu'un employé ou une association de l'école peut vous aider ?

— Vous pouvez utiliser une bibliothèque d'interface graphique pour passer d'un jeu en console à un jeu graphique.

— Vous pouvez changer des règles. Faites quelques changement drastiques, comme ajouter un nouveau type de cartes, des nouveaux effets, ajout d'aléatoire, ...

— Vous pouvez faire des annexes au jeu comme un éditeur de deck.

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.

2.5 Séquencement du travail

Voici, à titre indicatif les dates importantes du projet. Elles sont modulables selon le bon vouloir de votre chargé de projet. Chacun de ces rendus est séparé du suivant par deux semaines pour vous laisser le temps de corriger le Lot A et le Lot B si votre encadrant estime qu'il n'est pas correct ou qu'il est insuffisant pour pouvoir continuer avec le lot suivant. En particulier, puisque le Lot B nécessite la validation du A et que le C nécessite celle du B, les dates de rendus peuvent être amenées à changer.

Vous pouvez bien entendu rendre votre Lot A avant le 22 et le B avant le 26 si vous estimez avoir fini.