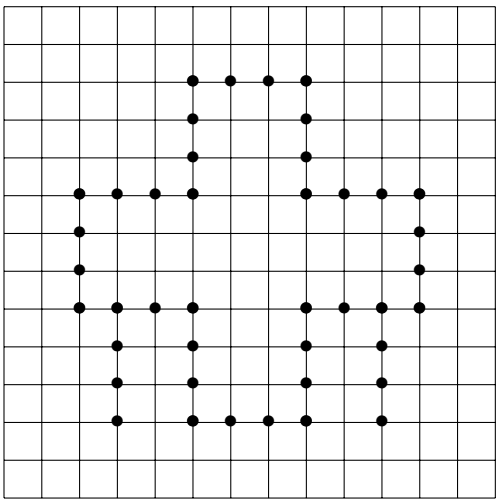
Projet informatique 2018-2019

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

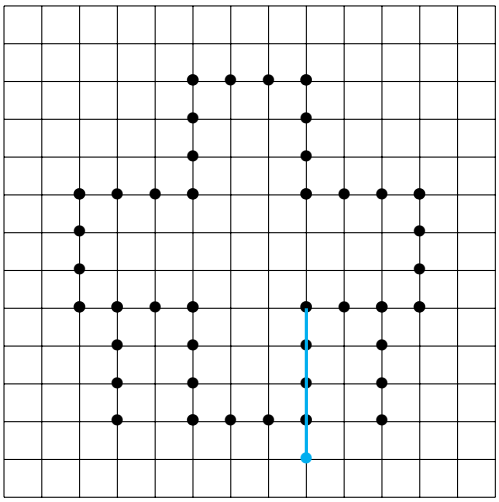
1.1 Le Morpion solitaire

On souhaite coder un jeux de solitaire nommé Morpion. Ce jeu se joue sur une grille de taille illimitée. Sur cette grille est initialement placé un nombre fini de points, par exemple comme sur la figure ci-après. On peut, bien entendu imaginer n'importe quelle configuration initiale. Elle n'est pas nécessairement symétrique.

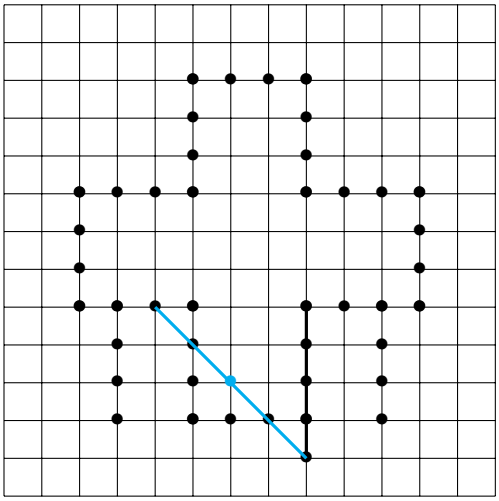


L'objectif est de rajouter des points sur la grille. Un point peut être rajouté à la grille s'il forme, avec 4 autres points, un alignement de cinq points adjacents horizontalement, verticalement ou en diagonale. On trace alors l'alignement sur la grille. Tout au long du jeux, deux alignements ne peuvent avoir plus d'un point en commun.

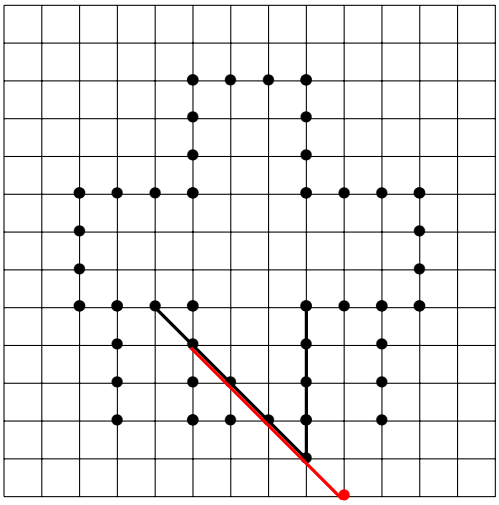
Dans l'exemple précédent, on peut rajouter le point ci dessous, représenté par un point bleu, formant un alignement avec les 4 points au dessus, indiqués par un segment bleu:



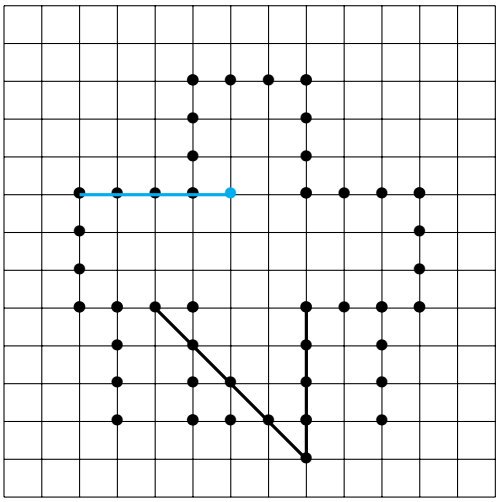
Il est possible, mais pas obligatoire, d'utiliser un point précédemment placé pour former un nouvel alignement. Par exemple, on peut ensuite placer le point suivant pour former un alignement incluant le point précédent.



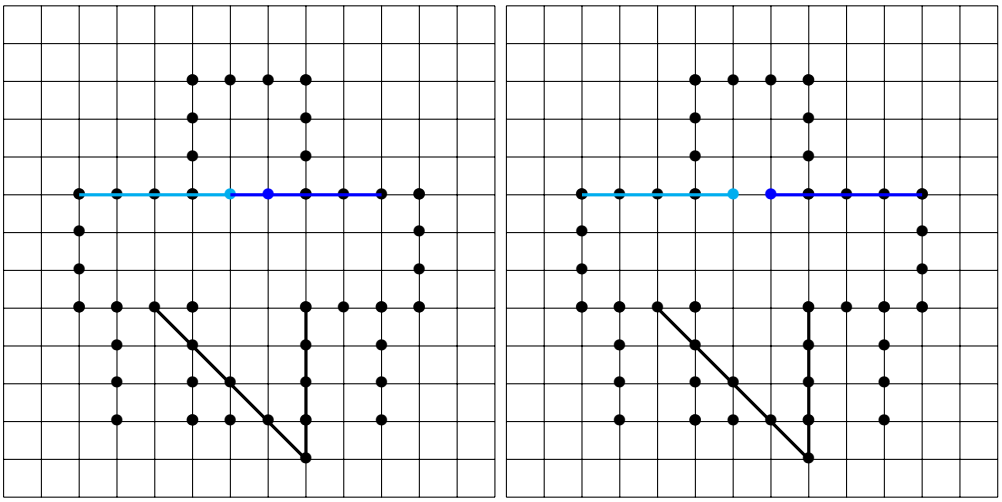
Il n'est pas possible de placer le point suivant, rouge, avec l'alignement rouge indiqué. En effet, l'alignement rouge et un des noirs auraient plus d'un point en commun.



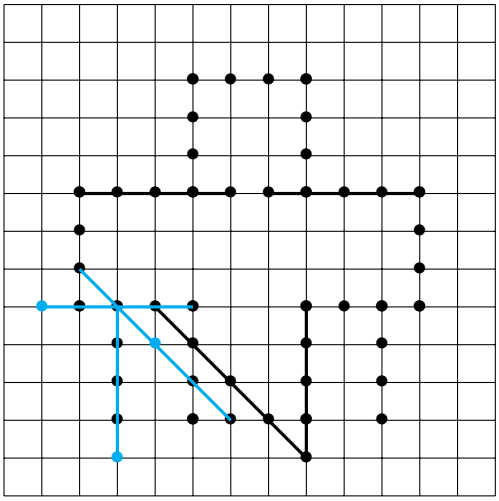
Par contre, si on ajoute le point suivant:



Alors on peut très bien ajouter, comme sur la figure de gauche, le point et l'alignement bleus foncés suivants. Les deux alignements n'ont bien qu'un seul point en commun. Petite remarque, comme la figure de droite le montre, deux alignements sont possibles lorsque l'on place le point bleu foncé, il faut en choisir un.



Enfin, on peut continer, par exemple, avec les points suivants. Un point peut être contenu dans autant d'alignements qu'on le souhaite.



On s'arrête quand plus aucun point ne peut être ajouté. Le but du jeux est d'ajouter le plus de points possibles.

1.2 Cahier des charges

On souhaite pouvoir jouer au morpion solitaire. Le jeu doit permettre de :

- Proposer de démarrer avec une grille préenregistrée dans un fichier;
- Jouer un coup (poser un point et définir l'alignement associé);
- Annuler le coup précédemement joué;
- Lister l'ensemble des coups possibles;
- Indiquer le score à chaque étape;
- Dire au joueur s'il est possible ou non de jouer un autre coup. Dans le cas contraire, le jeu s'arrête et le score s'affiche à l'écran.

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.

Deux outils n'ont pas nécessairement été présentés en cours: Doxygen et CUnit. Le premier permet de produire une documentation formatée et d'extraire cette documentation (pour qu'elle apparaisse par exemple sur une page web). Le second permet de faire des tests unitaires pour vérifier que vos fonctions ne contiennent pas (ou peu) de bugs. Ces deux étapes sont chronophages et donc moins prioritaires que les autres. Toutefois:

- votre code doit être commenté!
- votre code doit être testé!

Un code non commenté, un code qui ne compile pas, ou un code qui ne fonctionne pas se verra attribué la note de 0. Ce sont donc l'utilisation des outils et la méthodologie associée qui sont facultatives. Si elle n'a pas été faite en cours, une présentation de ces outils est prévue pendant les séances de projets par votre chargé de projet.

Le projet est découpé en 5 semaines: le 8 mars, le 22 mars, le 5 avril, le 19 avril, le 7 mai. Vous avez 3 rendus à faire: un premier rendu (lot A) la semaine du 8 mars, un autre (lot B) la semaine du 5 avril et un dernier (lot

C) la semaine du 7 mai. Chacun de ces rendus est séparé du suivant par une semaine 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.

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 le 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`. Ce code doit permettre, une fois compilé, 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 -c main.c`.

Remarque : en C, vous ne pouvez pas définir de structure vide. Pourtant, ce peut être utile si vous souhaitez générer un nouveau type, autre qu'un type primitif. Vous pouvez définir ces types avec, par exemple, `typedef struct _A {int i;} NT;` qui crée un nouveau type nommé `NT`.

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 implantation de sorte à jouer avec une interface graphique.

Tâches à réaliser pour le Lot A

1. Mettez en place le git et invitez votre encadrant sur votre dépôt. Ce git contiendra votre code. Créez une première branche `lot_a` dans laquelle vous déposerez le code du lot A. Vous devez maintenir votre dépot à jour, en poussant vos modifications au minimum une fois par semaine, à la fin de chaque séance (il est recommandé de pousser vos commits plus régulièrement).

2. Mettez en place un tableau Trello et invitez votre encadrant sur votre tableau. Ce tableau doit vous permettre de répartir les tâches (des Lots A, B, et C) entre vous, y compris celles-ci. Dans chaque tâche,

indiquez une (courte) description si le titre n'est pas explicite, une liste de sous-tâches si la tâche est divisible (notamment pour les parties de programmation) et, enfin, la liste des personnes affectées aux différentes tâches. Vous devez maintenir ce tableau à jour tout au long du projet au fur et à mesure que vous complétez des tâches.

3. Mettez en place les interfaces. Les interfaces sont au moins au nombre de 3, vous pouvez en produire d'autres si vous le jugez utile.

   - `plateau.h` est l'interface dont les fonctions permettent de gérer la grille, les points et les lignes qui sont dessus. Il contient des fonctions de: lecture d'un fichier pour produire un placement initial de points dans la grille; production d'un placement initial aléatoire ; écriture d'une grille, avec ses points dans un fichier; lister les coups valides; vérifier qu'un coup peut être joué; jouer un coup ; tester si la partie est terminée ou non.
   - `historique.h` est une surcouche du plateau enregistrant au fur et à mesure la liste des coups, permettant de les annuler et de les rejouer. Elle contient trois fonctions: jouer un coup sur un plateau ; annuler le dernier coup d'un plateau et rejouer le dernier coup annulé.
   - `interface.h` gère l'échange avec le joueur. Elle doit contenir trois fonctions : une fonction d'affichage du plateau, avec tous les coups joués, les alignements correspondants et, si demandé par l'utilisateur, la liste des coups possibles ; une fonction permettant au joueur de saisir un coup, de demander l'ensemble des coups disponibles, ou de demander comment fonctionne le logiciel ; et une fonction qui affiche une aide expliquant comment utiliser le logiciel.

4. Mettez en place un fichier `main.c` utilisant les fonctions et les types de `plateau.h`, `historique.h` et `interface.h` pour permettre à un joueur de jouer au morpion solitaire selon le cahier des charges de la section précédente. Le main ne doit contenir aucun appel des fonctions du mode console (en particulier printf et scanf). Ces appels seront faits dans l'implantation du module `interface.h`. Le main prendra en entrée deux arguments optionnels. Le premier est `-r` et s'utilisera ainsi: `main -r FILE` (où `main` est votre exécutable produit en sortie). Si cet

argument est présent, alors la grille est initialisée avec le contenu du fichier `FILE`. Sinon une grille par défault est générée (par exemple celle donnée en début de sujet). Le second est `-h` et s'utilisera ainsi: `main -h`. Si cet argument est présent, au lieu d'exécuter le programme, un texte d'aide s'affiche, expliquant à quoi sert cet exécutable et décrivant les 2 options `-h` et `-r`.

5. Mettez en place le makefile qui compilera le fichier `main.c` en un fichier `main.o` avec la commande `gcc -Wall -Wextra -c main.c`. Cette compilation doit s'effectuer sans erreur ni warning.

2.3 Lot B

Le second lot consiste à implanter les fonctions des différentes interfaces de sorte à pouvoir jouer au morpion solitaire en console. **Vous ne pouvez commencer ce lot que si le lot A a été validé par votre encadrant.**

Tâches à réaliser pour le Lot B

1. Créez une seconde branche `lot_b` sur votre dépôt git dans laquelle vous déposerez le code du lot B. Placez cette branche au niveau du commit pointé par la branche `lot_a`. Ne déplacez plus votre branche lot_a, vous pourriez avoir besoin de revenir sur cette branche ultérieurement pour le lot C.

2. Implanter toutes les corps de vos modules de sorte à ce que le joueur puisse jouer en console.

3. Complétez votre makefile de sorte à ce que le fichier `main.c` compile en un exécutable `main_console` qui permettra de jouer.

4. Tester votre code avec valgrind. Votre code doit s'exécuter sans erreur.

5. Facultatif : réécrivez votre documentation des interfaces au format Doxygen et produisez cette documentation sous forme de pages web.

2.4 Lot C

Le troisième lot consiste à implanter les fonctions des différentes interfaces de sorte à pouvoir jouer au morpion solitaire en mode graphique.

Vous ne pouvez commencer ce lot que si les lots A et B ont été validés par votre encadrant.

Pour fonctionner, une interface graphique crée généralement une boucle qui met à jour la fenêtre graphique en fonction des évènements qui surviennent à l'écran. Cette manière de faire est à priori incompatible avec votre fichier `main.c`. Pour y remédier et vous aider, une archive `gtk.tar.gz` vous est fournie. Cette archive contient une documentation, un Makefile et des fichiers de code d'exemple. L'exemple vous indique comment créer une fenêtre contenant un bouton et un dessin, comment dessiner un point ou une ligne sur la zone de dessin. Enfin, l'exemple vous montre comment, si vous cliquez sur le bouton ou le dessin, appeler une fonction et récupérer les informations utiles (notamment la position de la souris sur le dessin).

L'archive vous permettra d'adapter votre code pour générer un second exécutable qui lancera l'interface graphique et communiquera avec vos fonctions qui pourront ainsi envoyer ou recevoir des évènements (le joueur a cliqué ici, il a écrit ça, il faut dessiner un trait ici, un cercle là...). Toute l'interface est faite avec le module gtk. La documentation et l'exemple fournis avec l'archive vous indique comment, à partir de votre code, vous pouvez créer et manipuler l'interface. Le Makefile vous donnera un exemple pour pouvoir compiler votre code.

Enfin, attention, l'objectif n'est pas de remplacer le mode console par le mode graphique. Il faudra que votre projet crée 2 exécutables, un qui permette de jouer en mode console et un autre qui permette de jouer en mode graphique. Vous aurez donc deux implantations différentes du module `interface.h`. Cela signifie que, à l'issue du Lot C, votre module `interface.h` et votre fichier `main.c` devront être suffisamment abstraits pour permettre de jouer dans les deux modes.

Tâches à réaliser pour le Lot C

1. Créez une troisième branche `lot_c` sur votre dépôt git dans laquelle vous déposerez le code du lot C. Placez cette branche au niveau du commit pointé par la branche `lot_b`.

2. Copiez le fichier source du lot B relatif au module `interface.h`. Dans la suite, on désigne par `interfaceB.c` et `interfaceC.c` ces deux fichiers (mais ils peuvent avoir un autre nom dans votre projet).

3. Modifier votre fichier `interface.h`:

- Ajoutez un nouveau type abstrait correspondant à l'interface homme-machine utilisée (qui sera graphique ou console)
- Ajoutez deux fonctions permettant de créer et de détruire une interface.
- Modifiez les différentes signatures pour que les fonctions prennent en entrée l'interface.

4. Modifiez le fichier `main.c` en fonction des modifications apportées au fichier `interface.h`.

5. Modifiez le fichier `interfaceB.c` en fonction des modifications apportées au fichier `interface.h` et `main.c`. **Attention:** il ne faut pas modifier ce fichier dans la branche `lot_b`, il faut le modifier dans la branche `lot_c`. Il y a normalement assez peu de modifications à apporter par rapport au Lot B. En particulier, en mode console, une variable de type interface n'est pas nécessaire (puisqu'elle n'était pas nécessaire lors de la conception du Lot B). Vous pouvez donc simplement ignorer cette variable dans vos fonctions. Après cette étape, vous pouvez normalement compiler comme lors du Lot B.

6. Implantez le type abstrait et les fonctions du module `interface.h` dans le fichier `interfaceC.c` de sorte à ce qu'un joueur utilisant cette implantation joue avec une interface graphique.

7. Complétez votre makefile de sorte à ce que le fichier `main.c` puisse compiler en deux exécutables : `main_console` qui permettra de jouer en mode console et un autre, `main_gtk`, qui permettra de jouer en mode graphique.

8. Tester votre code avec valgrind. Votre code doit s'exécuter sans erreur.

9. Facultatif : Produisez un solveur qui trouvera à partir d'un instant donné une liste de coups maximisant le score final. Ajoutez à la boucle de jeu la possibilité de demander conseil. Ce conseil indiquera le prochain coup à jouer pour maximiser le score final.

10. Facultatif : sur la branche `lot_c` ou `lot_b` ; utilisez CUnit pour effectuer des tests unitaires sur vos implantations de l'interface `plateau.h`. Ces tests doivent justifier que les fonctions sont correctement implantées, sans erreur.

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 ; 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.