Stratego : nvs-06

nvs

16 juin 2022

Stratego : nvs-06 (56149 & 56172)

dépôt

https://git.esi-bru.be/56172/56149_stratego_56172

`git@git.esi-bru.be:56172/56149_stratego_56172.git`

.gitignore

il y a bien un fichier `.gitignore`, mais il n'a pas été utilisé dès le début : le répertoire `.git` fait 17,3 Mio.

modélisation

remise

tag / commit ok.

retard (void)

autre (void)

analyse

retour fait oralement.

console

remise

tag / commit
— ko : pas de `tag` ni de message de `commit` avec le texte `console release`.

retard (void)

autre (void)

documentation

— les fichiers ne sont pas documentés => pas de documentation produite pour les énumérations `LEVEL`, `DIRECTION`, `PLAYER`, `PIECE` et `STATUS`.
— pas de documentation des fonctions / opérateurs `std::ostream& operator<<(std::ostream& os, GameElement const & dt)` et `std::ostream& operator<<(std::ostream& os, const Position& pos)`.
— pas de documentation pour les destructeurs d'`Observer` et d'`Observable`.
— pour le reste, les méthodes publiques et privées sont bien documentées.

fichier de configuration uniquement ok.

à l'ouverture de celui-ci avec doxywizard (doxygen 1.9.1), j'obtiens :

```
warning: ignoring unsupported tag 'SHOW_HEADERFILE' at line 609, file C:/.../nvs-06_56149_56172/Doxyfile
warning: ignoring unsupported tag 'WARN_IF_INCOMPLETE_DOC' at line 827, file C:/.../nvs-06_56149_56172/Doxyfile
warning: ignoring unsupported tag 'DOCSET_FEEDURL' at line 1410, file C:/.../nvs-06_56149_56172/Doxyfile
warning: ignoring unsupported tag 'FULL_SIDEBAR' at line 1621, file C:/.../nvs-06_56149_56172/Doxyfile
warning: ignoring unsupported tag 'OBFUSCATE_EMAILS' at line 1652, file C:/.../nvs-06_56149_56172/Doxyfile
warning: ignoring unsupported tag 'MATHJAX_VERSION' at line 1711, file C:/.../nvs-06_56149_56172/Doxyfile
warning: ignoring unsupported tag 'DIR_GRAPH_MAX_DEPTH' at line 2568, file C:/.../nvs-06_56149_56172/Doxyfile
```

lorsque je compile, j'obtiens :

```
warning: tag INPUT: input source 'C:/Users/Administrator/56149_stratego_56172/Stratego' does not exist
Doxygen version used: 1.9.1 (ef9b20ac7f8a8621fcfc299f8bd0b80422390f4b)
error: tag OUTPUT_DIRECTORY: Output directory 'C:/Users/Administrator/56149_stratego_56172' does not exist and cannot be created
```

```
Exiting...
*** Doxygen has finished
```

je modifie le fichier de configuration avec des chemins *relatifs* pour les dossiers source et destination et ça passe avec quand même une erreur :

```
error: Project logo 'C:/Users/Administrator/Desktop/communityIcon_i69s9qgzn1541.png' specified by PROJECT_LOGO does not exist!
lookup cache used 230/65536 hits=912 misses=245
```

le fichier `communityIcon_i69s9qgzn1541.png` ne se trouve pas dans le dépôt...

rapport

format pdf ok.

bogue non signalé (void)

écart / ajout non signalé (void)

autre (void)

rapport / code

avertissement restant

gcc signalé

(void)

non signalé

(void)

gcc + clang-analyzer (void)

clang++ (void)

clang++ + clang-analyzer (void)

cppcheck signalé

```
model/board.cpp: 114:16 [model/board.cpp:110] -> [model/board.cpp:114]
style : knownConditionTrueFalse
        if(row >= 0 && row <= 3){
                ^
Condition 'row>=0' is always true
```

à régler

```
model/game.cpp: 111:45 [model/game.cpp:111]
style : constParameter
optional<GameElement> Game::move(Position & p, DIRECTION & d, int distance){
                                             ^
Parameter 'p' can be declared with const
```

à régler

non signalé

```
model/game.h: 32:10 [model/game.cpp:49] -> [model/game.h:32]
style (inconclusive) : functionConst
    bool checkCanMakeMove(std::vector<std::string> &);
         ^
Technically the member function 'Game::checkCanMakeMove' can be const.
```

à régler

```
model/game.h: 162:10 [model/game.cpp:261] -> [model/game.h:162]
performance (inconclusive) : functionStatic
    bool goodFile(std::vector<std::string> const & pieces);
         ^
Technically the member function 'Game::goodFile' can be static (but you may consider moving to unnamed namespace).
```

à régler éventuellement

```
model/gameElement.h: 30:5 [model/gameElement.h:30]
```

```
style : noExplicitConstructor
    GameElement(std::string const & decor);
    ^
Class 'GameElement' has a constructor with 1 argument that is not explicit.
```

à régler éventuellement

```
model/game.h: 60:5 [model/game.h:60]
style : noExplicitConstructor
    Game(Board & board, LEVEL level = DEBUTANT);
    ^
Class 'Game' has a constructor with 1 argument that is not explicit.
```

à régler éventuellement

```
model/board.h: 135:20 [model/board.h:137] -> [model/board.h:135]
style (inconclusive) : duplicateBranch
            } else if(bd.getSoldier(Position(i,j)).getDecor() == "W"){
                   ^
Found duplicate branches for 'if' and 'else'.
```

à régler

```
nofile: 0:0
information : missingInclude

Cppcheck cannot find all the include files (use --check-config for details)
```

?

code source

portabilité

casse noms fichiers ok.

séparateur / ok.

c++ standard ok.

si pas std : portabilité (void)

bonnes pratiques

déclarations anticipées si possible

— dans `position.h` :

```
// #include <iostream>  // rnvs : include / déclaration anticipée
#include <string>       // rnvs : include / déclaration anticipée
#include <ostream>      // rnvs : include / déclaration anticipée
```

— dans `game.h` :

```
// #include "board.h"       // rnvs : include / déclaration anticipée

#include "enum.h"           // rnvs : include / déclaration anticipée
#include <optional>         // rnvs : include / déclaration anticipée
#include <vector>           // rnvs : include / déclaration anticipée
#include <string>           // rnvs : include / déclaration anticipée

class Board;                // rnvs : include / déclaration anticipée
class GameElement;          // rnvs : include / déclaration anticipée
class Position;             // rnvs : include / déclaration anticipée
```

— dans `board.h` :

```
#include "enum.h"           // rnvs : include / déclaration anticipée
```

— dans `view.h` :

```
// #include "model/game.h"      // rnvs : include / déclaration anticipée
// #include "keyboardAndStringConvert/keyboard.hpp" // rnvs : include / déclaration anticipée
#include <optional>     // rnvs : include / déclaration anticipée

class Game;         // rnvs : include / déclaration anticipée
class GameElement;  // rnvs : include / déclaration anticipée
class Position;     // rnvs : include / déclaration anticipée
```

using namespace dans .h ok.

autre (void)

gestion de la mémoire ok : pas de `new` dans les classes métier ni celles de l'application console.

tests unitaires (ceci n'est pas demandé)

(void)

classes métier

initialisation plateau

ok : plateau de jeu construit vide avec les pièces d'eau bien positionnées.

interactive

le travail est bien réalisé, mais ko :

- explosé entre contrôleur et métier
- voir `Controller::initialiseGamePlacement(PLAYER currentPlayer)` et `Controller::fillVectorSoldier(vector<GameElement> & soldier, PLAYER color)`.
- voir `Game::initializeBattleField(const Position & p, PLAYER player, const GameElement & s)`.

fichiers

ok : voir `Game::initializeBattelField(const vector<string> & file)` et `Board::initializeBattleField(vector<string> s, PLAYER player)`.

ko :

- c'est le même fichier qui est utilisé pour les 2 joueurs (voir `Game::initializeBattelField(const vector<string> & file)`), contrairement à ce qui est demandé dans l'énoncé.

validation du nombre de pièces de chaque type

- interactif : ko : `Board::canBePlaceStart(Position p, PLAYER player)` ne vérifie que si le placement se fait sur les bonnes lignes et pas sur de l'eau. le contrôle du nombre se fait dans `Controller::initialiseGamePlacement(PLAYER currentPlayer)` et `Controller::fillVectorSoldier(vector<GameElement> & soldier, PLAYER color)`, il n'est pas réalisé par les classes métier, mais le contrôleur.

— fichiers : ko : on a bien une méthode du modèle `Game::goodFile(vector<string> const & pieces)` mais elle n'est pas appelée automatiquement par `Game::initializeBattelField(const vector<string> & file)` mais par le contrôleur dans `Controller::initialiseGameFile()`.

possibilité de mode débutant

ok : `Game::setLevel(LEVEL level)` uniquement utilisable quand le jeu est dans l'état `PLACEMENT`.

joueurs (éventuellement)

(void)

déplacement obligatoire

ok : `Game::move(Position & p, DIRECTION & d, int distance)` vérifie que le jeu est dans l'état `MOVE` et change cet état si le mouvement a bien eu lieu.

ko :

— je ne vois pas où dans les méthodes métier on vérifie que la position de départ contient bien un pion du joueur actif.

pièces d'eau inaccessibles

ok : `Board::canBePlace(const Position & p)` appelé par méthode métier de déplacement.

éclaireur

latéral uniquement

ok : `Game::move(Position & p, DIRECTION & d, int distance)` utilise une direction parmi `GAUCHE`, `DROITE`, `HAUT`et `BAS`.

illimité sauf obstacle

ok : `Board::moveEclaireur(Position & p, Position & toGo, Position dir, int distance)` utilise itérativement `Board::canBePlace(const Position & p)` qui le vérifie.

bombes et drapeau immobiles

ok :

— `Board::move(Position p, DIRECTION dir, int distance)` le vérifie.
— *rem.* : `Controller::conditionPosition(Position pos, Position toGo)` se charge également de le vérifier.

autres pièces

latéral uniquement

ok : `Game::move(Position & p, DIRECTION & d, int distance)` utilise une direction parmi `GAUCHE`, `DROITE`, `HAUT`et `BAS`.

une seule position

ok : `Board::move(Position p, DIRECTION dir, int distance)` le vérifie (`if (distance > 1 && getSoldier(p).getRole() == ECLAIREUR)`).

limite des 3 allers - retours

ok : voir `Game::move(Position & p, DIRECTION & d, int distance)`, `Game::fillMovementList(bool & canMakeMove, Position toGo)` et `Game::checkCanMakeMove(vector<string> & movement)`.

combat détection

ok : voir `Board::move(Position p, DIRECTION dir, int distance)`.

résolution

ok : voir `Board::attack(Position p , Position toAttack)`.

cas de l'espionne

ok : voir `Board::attack(Position p , Position toAttack)`.

fin de partie ok : `Board::endGame()` invoqué par `Game::endGame()` elle-même par `Game::move(Position & p, DIRECTION & d, int distance)`.

par prise de drapeau

ok : voir `Board::endGame()`.

par impossibilité de déplacement

ok : voir `Board::endGame()`.

ko :

— dans `Board::endGame()` on vérifie s'il reste des pions déplaçables *en théorie*, mais il n'y a pas de vérification si effectivement ils sont déplaçables.

méthodes complètes : 1 méthode / 1 action de jeu

ko :

- le placement interactif est *splitté* entre `Controller` et `Game` (voir ci-dessus).
- le placement via fichier est *splitté* entre `Controller` et `Game` (voir ci-dessus).
- pour ce qui concerne le déplacement, le contrôleur réalise des vérifications qui sont réalisées à nouveau dans les méthodes métier. c'est donc ok pour les déplacements et combats.

impossibilité de tricher (bibliothèque)

totalement ko :

- le `Board` est fourni par référence au `Game`. il est dès lors possible d'y accéder en toute liberté sans aucun contrôle des méthodes de `Game` et de l'état du jeu. c'est dommage car dans `Game`, il y a utilisation d'un état (sans *setter*) pour bien verrouiller l'appel des méthodes publiques et que le *getter* optional Game : :getSoldier(Position p) retourne des clones des éléments de jeu.
  rem. : contrairement à ce qu'il semble, il n'y a pas de *getter* de `Board` implémenté.
  le problème, c'est qu'on a accès au plateau directement sur celui ;fourni au constructeur de `Game`. il suffirait très probablement de cloner le `Board` lors de la construction du `Game`...
-

contrôleur

fiabilisation lectures clavier ok.

respect de la convention d'identification des cases ok.

légende

ok.

convivialité

- il est assez pénible de devoir taper `BAS`, `HAUT`, `GAUCHE`, `DROITE` : l'initiale pourrait suffire. c'est la même chose pour indiquer le type de jeu et d'initialisation.
- pas de détection de sélection d'une pièce déplaçable : il faut fournir une direction.

alternance des joueurs plateau caché entre les joueurs

ok : défilement vertical de l'affichage.

information si un combat a eu lieu

ok : uniquement en mode normal. en mode débutant, on voit le pions adverse. en cas d'égalité, c'est un peu moins évident de savoir où et si un combat a eu lieu.

vue

design pattern observer ko :

- il y a des classes `Observable` et `Observer`, mais elles ne sont pas utilisées.
- la classe `View` n'est pas un observateur, la classe `Game` n'est pas observable.
- c'est dans les méthodes de la classe `Controller` que les méthodes de `View` et `Game` sont explicitement invoquées => pas de notification automatique, donc pas réellement de vue attachée à un modèle => MVC pas ok.

absence de flux (cout) dans classes métier ok.

affichage masqué en mode normal

ok.

éventuellement non masqué en mode débutant

ok.

cimetière

pas mis en œuvre.

autre

- pas de fichier de configuration de plateau fourni.

gui

remise

tag / commit
- pas de *tag* `gui`.
- plusieurs *commits* où `gui` apparaît, mais pas `gui release`.

retard (void)

autre (void)

documentation

- pas de fichier pour doxygen.
- pas de documentation fournie.
- les fichiers et classes graphiques ne sont pas documentés. les méthodes publiques et privées des classes graphiques sont partiellement documentées.

rapport

format pdf ok.

bogue non signalé (void)

écart / ajout non signalé (void)

autre (void)

code source

portabilité

casse noms fichiers ok.

séparateur / ok.

c++ standard + qt ok.

si pas std + qt : portabilité (void)

gestion de la mémoire (void)

contrôleur

respect des règles ko :

— un seul fichier lu : les 2 joueurs utilisent le même en miroir.

convivialité

— placement interactif propre et facile : ok.
— choix des fichiers avec `QFileDialog` : ok.
— choix d'une direction avec touches fléchées (on peut changer tant qu'on veut), puis du pion à déplacer : c'est moyennement convivial mais je suppose qu'on s'y habitue.

vue

design pattern observer ok.

convivialité

— image et valeurs des pions : ok.
— rapport de combat en mode normal.
— pas de masquage du plateau entre les 2 joueurs.
— *rem.* : sous gnu / linux l'image de fond et celle de l'eau disparaît après placement des pions (testé uniquement pour lecture de fichier).

autre

— pas de fichier de placement initial fourni.

examen

retours faits le jour même.