Diaballik : n15_48502_49880

nvs

23 juin 2020

Diaballik : n15_48502_49880

console

remise

retard (void)

autre (void)

documentation

— fichiers pas documentés avec `\file`
— pas de documentation pour les méthodes des classes `Observable`, `Observer`

rapport

format pdf (void)

bogue non signalé

— dans `Board::initBoard()` si plateau pas carré : plantage
— balles pas bien placées si pas 7 x 7 : voir `Board::initBoard()`
— supports pas bien placés si pas 7 x 7 : voir `Board::initBoard()`
— pour tester les bogues ci-dessus non signalés : changer la taille du plateau dans l'appel du constructeur de `Game` dans `main.cpp`

— les seules passes possibles sont en diagonales... et elles passent par dessus un support adverse : ko
— plantage en fin de partie quand balle sur ligne adverse :

```
the winner is : z

pure virtual method called
terminate called without an active exception
```

écart / ajout non signalé
— possibilité d'avoir plateau non carré !

autre (void)

rapport / code

avertissement restant

gcc signalé

(void)

non signalé

```
Board.cpp: Dans la fonction membre « bool BoardSpace::Board::checkMove(int, int, int, int) »:
Board.cpp:81:10: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
   81 |     if(ox>=0 & ox<width_ & oy >= 0 & oy <height_ & dx>=0 & dx <width_ & dy >=0 & dy < height_){
      |        ~~^~~
Board.cpp:81:31: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
   81 |     if(ox>=0 & ox<width_ & oy >= 0 & oy <height_ & dx>=0 & dx <width_ & dy >=0 & dy < height_){
      |                           ~~~^~~~
Board.cpp:81:41: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
   81 |     if(ox>=0 & ox<width_ & oy >= 0 & oy <height_ & dx>=0 & dx <width_ & dy >=0 & dy < height_){
      |                                     ~~~^~~~~~~~
Board.cpp:81:54: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
   81 |     if(ox>=0 & ox<width_ & oy >= 0 & oy <height_ & dx>=0 & dx <width_ & dy >=0 & dy < height_){
      |                                                      ~~^~~
```

```
Board.cpp:81:63: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
   81 |     if(ox>=0 & ox<width_ & oy >= 0 & oy <height_ & dx>=0 & dx <width_ & dy >=0 & dy < height_){
      |                                                       ~~~^~~~~~~
Board.cpp:81:76: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
   81 |     if(ox>=0 & ox<width_ & oy >= 0 & oy <height_ & dx>=0 & dx <width_ & dy >=0 & dy < height_){
      |                                                                    ~~~^~~
Board.cpp:81:85: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
   81 |     if(ox>=0 & ox<width_ & oy >= 0 & oy <height_ & dx>=0 & dx <width_ & dy >=0 & dy < height_){
      |                                                                          ~~~^~~~~~~~~
Board.cpp:82:15: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
   82 |         if((dx==ox-1 & dy ==oy) || (dx==ox & dy==oy+1) || (dx==ox+1 & dy==oy) || (dx==ox & dy==oy-1)){
      |             ~~^~~~~~
Board.cpp:82:39: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
   82 |         if((dx==ox-1 & dy ==oy) || (dx==ox & dy==oy+1) || (dx==ox+1 & dy==oy) || (dx==ox & dy==oy-1)){
      |                                       ~~^~~~
Board.cpp:82:62: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
   82 |         if((dx==ox-1 & dy ==oy) || (dx==ox & dy==oy+1) || (dx==ox+1 & dy==oy) || (dx==ox & dy==oy-1)){
      |                                                       ~~^~~~~~
Board.cpp:82:85: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
   82 |         if((dx==ox-1 & dy ==oy) || (dx==ox & dy==oy+1) || (dx==ox+1 & dy==oy) || (dx==ox & dy==oy-1)){
      |                                                                           ~~^~~~
Board.cpp: Dans la fonction membre « bool BoardSpace::Board::checkPasse(int, int, int, int, Color) »:
Board.cpp:125:10: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
  125 |     if(ox<dx & oy < dy){
      |        ~~^~~
Board.cpp:129:10: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
  129 |     if(ox<dx & oy>dy){
      |        ~~^~~
Board.cpp:133:10: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
  133 |     if(ox>dx & oy>dy){
      |        ~~^~~
Board.cpp:137:10: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
  137 |     if(ox>dx & oy<dy){
      |        ~~^~~
Board.cpp: Dans la fonction membre « PieceSpace::Piece BoardSpace::Board::direction(int, int, int, int, Color) »:
```

```
Board.cpp:151:26: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
  151 |     while(directionWidth >= 0 & directionWidth < width_ & directionHeight >= 0 & directionHeight < height_
      |           ~~~~~~~~~~~~~~~~^~~~
Board.cpp:151:75: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
  151 |     while(directionWidth >= 0 & directionWidth < width_ & directionHeight >= 0 & directionHeight < height_
      |                                                           ~~~~~~~~~~~~~~~~^~~~
Board.cpp:151:98: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
  151 |     while(directionWidth >= 0 & directionWidth < width_ & directionHeight >= 0 & directionHeight < height_
      |                                                                                 ~~~~~~~~~~~~~~~~^~~~~~~~
Board.cpp:152:65: attention: parenthèses suggérées autour de la comparaison dans l'opérande de « & » [-Wparentheses]
  152 |             & board_[directionWidth][directionHeight].getColor()!=color){
      |               ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~
```

à régler éventuellement

gcc + clang-analyzer idem gcc

clang++ (void)

clang++ + clang-analyzer idem gcc

cppcheck signalé

(void)

non signalé

```
Piece.h:35:5: style: Class 'Piece' has a constructor with 1 argument that is not explicit. [noExplicitConstructor]
    Piece (Color color);
    ^
```

à régler éventuellement

```
Game.cpp:15:16: style:inconclusive: Function 'Game' argument 1 names different: declaration 'x' definition 'width'. [funcArgNamesDifferen
Game::Game(int width, int height):
               ^
```

à régler éventuellement

```
Game.cpp:15:27: style:inconclusive: Function 'Game' argument 2 names different: declaration 'y' definition 'height'. [funcArgNamesDiffere
Game::Game(int width, int height):
                          ^
```

à régler éventuellement

```
Board.cpp:81:14: style:inconclusive: Boolean expression 'ox>=0' is used in bitwise operation. Did you mean '&&'? [bitwiseOnBoolean]
    if(ox>=0 & ox<width_ & oy >= 0 & oy <height_ & dx>=0 & dx <width_ & dy >=0 & dy < height_){
             ^
```

à régler éventuellement

```
Board.cpp:81:26: style:inconclusive: Boolean expression 'oy>=0' is used in bitwise operation. Did you mean '&&'? [bitwiseOnBoolean]
    if(ox>=0 & ox<width_ & oy >= 0 & oy <height_ & dx>=0 & dx <width_ & dy >=0 & dy < height_){
                          ^
```

à régler éventuellement

```
Board.cpp:81:36: style:inconclusive: Boolean expression 'oy<height_' is used in bitwise operation. Did you mean '&&'? [bitwiseOnBoolean]
    if(ox>=0 & ox<width_ & oy >= 0 & oy <height_ & dx>=0 & dx <width_ & dy >=0 & dy < height_){
                                   ^
```

à régler éventuellement

```
Board.cpp:81:50: style:inconclusive: Boolean expression 'dx>=0' is used in bitwise operation. Did you mean '&&'? [bitwiseOnBoolean]
    if(ox>=0 & ox<width_ & oy >= 0 & oy <height_ & dx>=0 & dx <width_ & dy >=0 & dy < height_){
                                                 ^
```

à régler éventuellement

```
Board.cpp:81:58: style:inconclusive: Boolean expression 'dx<width_' is used in bitwise operation. Did you mean '&&'? [bitwiseOnBoolean]
    if(ox>=0 & ox<width_ & oy >= 0 & oy <height_ & dx>=0 & dx <width_ & dy >=0 & dy < height_){
                                                           ^
```

à régler éventuellement

```
Board.cpp:81:71: style:inconclusive: Boolean expression 'dy>=0' is used in bitwise operation. Did you mean '&&'? [bitwiseOnBoolean]
    if(ox>=0 & ox<width_ & oy >= 0 & oy <height_ & dx>=0 & dx <width_ & dy >=0 & dy < height_){
                                                                      ^
```

à régler éventuellement

```
Board.cpp:81:80: style:inconclusive: Boolean expression 'dy<height_' is used in bitwise operation. Did you mean '&&'? [bitwiseOnBoolean]
    if(ox>=0 & ox<width_ & oy >= 0 & oy <height_ & dx>=0 & dx <width_ & dy >=0 & dy < height_){
                                                                               ^
```

à régler éventuellement

```
Board.cpp:82:22: style:inconclusive: Boolean expression 'dx==ox-1' is used in bitwise operation. Did you mean '&&'? [bitwiseOnBoolean]
        if((dx==ox-1 & dy ==oy) || (dx==ox & dy==oy+1) || (dx==ox+1 & dy==oy) || (dx==ox & dy==oy-1)){
                     ^
```

à régler éventuellement

```
Board.cpp:82:44: style:inconclusive: Boolean expression 'dx==ox' is used in bitwise operation. Did you mean '&&'? [bitwiseOnBoolean]
        if((dx==ox-1 & dy ==oy) || (dx==ox & dy==oy+1) || (dx==ox+1 & dy==oy) || (dx==ox & dy==oy-1)){
                                           ^
```

à régler éventuellement

```
Board.cpp:82:69: style:inconclusive: Boolean expression 'dx==ox+1' is used in bitwise operation. Did you mean '&&'? [bitwiseOnBoolean]
        if((dx==ox-1 & dy ==oy) || (dx==ox & dy==oy+1) || (dx==ox+1 & dy==oy) || (dx==ox & dy==oy-1)){
                                                                    ^
```

à régler éventuellement

```
Board.cpp:125:14: style:inconclusive: Boolean expression 'ox<dx' is used in bitwise operation. Did you mean '&&'? [bitwiseOnBoolean]
    if(ox<dx & oy < dy){
             ^
```

à régler éventuellement

```
Board.cpp:129:14: style:inconclusive: Boolean expression 'ox<dx' is used in bitwise operation. Did you mean '&&'? [bitwiseOnBoolean]
    if(ox<dx & oy>dy){
             ^
```

à régler éventuellement

```
Board.cpp:133:14: style:inconclusive: Boolean expression 'ox>dx' is used in bitwise operation. Did you mean '&&'? [bitwiseOnBoolean]
    if(ox>dx & oy>dy){
             ^
```

à régler éventuellement

```
Board.cpp:137:14: style:inconclusive: Boolean expression 'ox>dx' is used in bitwise operation. Did you mean '&&'? [bitwiseOnBoolean]
    if(ox>dx & oy<dy){
             ^
```

à régler éventuellement

```
Board.cpp:151:31: style:inconclusive: Boolean expression 'directionWidth>=0' is used in bitwise operation. Did you mean '&&'? [bitwiseOnB
    while(directionWidth >= 0 & directionWidth < width_ & directionHeight >= 0 & directionHeight < height_
                              ^
```

à régler éventuellement

```
Board.cpp:151:57: style:inconclusive: Boolean expression 'directionHeight>=0' is used in bitwise operation. Did you mean '&&'? [bitwiseOn
    while(directionWidth >= 0 & directionWidth < width_ & directionHeight >= 0 & directionHeight < height_
                                                       ^
```

à régler éventuellement

```
Board.cpp:151:80: style:inconclusive: Boolean expression 'directionHeight<height_' is used in bitwise operation. Did you mean '&&'? [bitw
    while(directionWidth >= 0 & directionWidth < width_ & directionHeight >= 0 & directionHeight < height_
                                                                              ^
```

à régler éventuellement

```
Board.cpp:152:13: style:inconclusive: Boolean expression 'board_[directionWidth][directionHeight].getColor()!=color' is used in bitwise o
           & board_[directionWidth][directionHeight].getColor()!=color){
           ^
```

à régler éventuellement

```
Board.h:58:11: style:inconclusive: Technically the member function 'BoardSpace::Board::getPiece' can be const. [functionConst]
    Piece getPiece(int x, int y);
          ^
```

à régler

```
Board.h:87:9: style:inconclusive: Technically the member function 'BoardSpace::Board::getWidth' can be const. [functionConst]
    int getWidth();
        ^
```

à régler

```
Board.h:93:9: style:inconclusive: Technically the member function 'BoardSpace::Board::getHeight' can be const. [functionConst]
    int getHeight();
        ^
```

à régler

```
Board.h:105:10: style:inconclusive: Technically the member function 'BoardSpace::Board::checkMove' can be const. [functionConst]
    bool checkMove(int ox,int oy, int dx, int dy);
         ^
```

à régler

```
Board.cpp:81:14: style: Boolean result is used in bitwise operation. Clarify expression with parentheses. [clarifyCondition]
    if(ox>=0 & ox<width_ & oy >= 0 & oy <height_ & dx>=0 & dx <width_ & dy >=0 & dy < height_){
              ^
```

à régler éventuellement

```
Board.cpp:82:22: style: Boolean result is used in bitwise operation. Clarify expression with parentheses. [clarifyCondition]
        if((dx==ox-1 & dy ==oy) || (dx==ox & dy==oy+1) || (dx==ox+1 & dy==oy) || (dx==ox & dy==oy-1)){
                     ^
```

à régler éventuellement

```
Board.cpp:125:14: style: Boolean result is used in bitwise operation. Clarify expression with parentheses. [clarifyCondition]
    if(ox<dx & oy < dy){
              ^
```

à régler éventuellement

```
Board.cpp:129:14: style: Boolean result is used in bitwise operation. Clarify expression with parentheses. [clarifyCondition]
    if(ox<dx & oy>dy){
              ^
```

à régler éventuellement

```
Board.cpp:133:14: style: Boolean result is used in bitwise operation. Clarify expression with parentheses. [clarifyCondition]
    if(ox>dx & oy>dy){
              ^
```

à régler éventuellement

```
Board.cpp:137:14: style: Boolean result is used in bitwise operation. Clarify expression with parentheses. [clarifyCondition]
    if(ox>dx & oy<dy){
             ^
```

à régler éventuellement

```
Board.cpp:151:31: style: Boolean result is used in bitwise operation. Clarify expression with parentheses. [clarifyCondition]
    while(directionWidth >= 0 & directionWidth < width_ & directionHeight >= 0 & directionHeight < height_
                              ^
```

à régler éventuellement

```
Board.cpp:152:13: style: Boolean result is used in bitwise operation. Clarify expression with parentheses. [clarifyCondition]
            & board_[directionWidth][directionHeight].getColor()!=color){
            ^
```

à régler éventuellement

```
Board.cpp:67:56: style:inconclusive: Function 'move' argument 5 names different: declaration 'color' definition 'currentColor'. [funcArgN
void Board::move(int ox, int oy, int dx, int dy, Color currentColor){
                                                       ^
```

à régler éventuellement

```
Board.cpp:156:12: warning: Either the condition 'directionWidth>=0' is redundant, otherwise there is negative array index -1. [negativeCo
    return board_[directionWidth][directionHeight];
           ^
Board.cpp:151:26: note: Assuming that condition 'directionWidth>=0' is not redundant
    while(directionWidth >= 0 & directionWidth < width_ & directionHeight >= 0 & directionHeight < height_
                         ^
Board.cpp:156:12: note: Negative array index
    return board_[directionWidth][directionHeight];
           ^
```

à régler éventuellement

```
Piece.h:42:10: style:inconclusive: Technically the member function 'PieceSpace::Piece::isInside' can be const. [functionConst]
    bool isInside();
         ^
```

à régler

```
Piece.h:56:11: style:inconclusive: Technically the member function 'PieceSpace::Piece::getColor' can be const. [functionConst]
    Color getColor();
          ^
```

à régler

```
Piece.h:63:10: style:inconclusive: Technically the member function 'PieceSpace::Piece::getBool' can be const. [functionConst]
    bool getBool();
         ^
```

à régler

```
Players.h:44:12: style:inconclusive: Technically the member function 'PlayersSpace::Players::getName' can be const. [functionConst]
    string getName();
           ^
```

à régler

```
Players.h:57:9: performance:inconclusive: Technically the member function 'PlayersSpace::Players::play' can be static (but you may consid
    int play(istream & c);
        ^
```

à régler éventuellement

```
Players.h:64:11: style:inconclusive: Technically the member function 'PlayersSpace::Players::getColor' can be const. [functionConst]
    Color getColor();
          ^
```

à régler

```
Players.cpp:11:25: performance: Function parameter 'name' should be passed by const reference. [passedByValue]
Players::Players(string name,Color color):
                         ^
```

à régler

```
View.h:67:10: performance:inconclusive: Technically the member function 'ViewSpace::View::showCommand' can be static (but you may conside
    void showCommand();
         ^
```

à régler éventuellement

```
View.h:75:10: performance:inconclusive: Technically the member function 'ViewSpace::View::checkAction' can be static (but you may conside
    bool checkAction(string action);
         ^
```

à régler éventuellement

```
View.h:39:5: style: Class 'View' has a constructor with 1 argument that is not explicit. [noExplicitConstructor]
    View(Game & game);
    ^
```

à régler éventuellement

```
View.cpp:89:30: performance: Function parameter 'action' should be passed by const reference. [passedByValue]
void View::playAction(string action,int & countMove,int & countPasse){
                              ^
```

à régler

code source

portabilité

casse noms fichiers (void)

séparateur / (void)

c++ standard (void)

si pas std : portabilité (void)

bonnes pratiques

déclarations anticipées si possible

- `#include "Players.h"` inutile dans `Board.h` : déclaration anticipée est suffisante
- `#include "Observers.h"` inutile dans `Observable.h` : déclaration anticipée est suffisante

— `#include "Game.h"` inutile dans `View.h` : déclaration anticipée est suffisante

using namespace dans .h `using namespace std;` dans :

— `Board.h`
— `Observable.h`
— `Piece.h`
— `Players.h`

`using namespace PlayersSpace;` dans :

— `Board.h`
— `Game.h`

`using namespace BoardSpace;` dans :

— `Game.h`

`using namespace ObservableSpace;` dans :

— `Game.h`
— `Observable.h`

`using namespace PieceSpace;` dans :

— `Board.h`

autre

— `#include <iostream>` inutile dans `Piece.h`
— `#include <iostream>` inutile dans `Piece.cpp`
— `#include <string>` manquant dans `Piece.cpp`
— `#include <vector>` inutile dans `Game.h`
— `#include <ostream>` manquant dans `Game.h`
— `#include <string>` manquant dans `View.h`
— `#include <ostream>` manquant dans `Board.cpp`

gestion de la mémoire ok

classes métier

initialisation plateau

- pas de vérification que le plateau est 5 x 5 ou 7 x 7 ou 9 x 9
- possibilité d'avoir plateau rectangulaire !
- plantage dans `Board::initBoard()` si plateau pas carré : ko !

supports

- ko : supports pas bien placée si pas 7 x 7 : voir `Board::initBoard()`

balles

- ko : balle pas bien placée si pas 7 x 7 : voir `Board::initBoard()`

possibilité de variante

- ko : pas implémenté

joueurs (éventuellement)

(void)

déplacement support latéral uniquement

ok

d'une seule position

ok

impossible de déplacer support avec balle

ok

un seul support par emplacement

ok

maximum 2 déplacements par tour de jeu

- ko : la méthode `Game::move(int ox, int oy, int dx, int dy)` ne tient pas à jour un compteur de déplacement : c'est géré dans le contrôleur

lancer balle latéral ou diagonal

— ko : les seules passes que j'arrive à faire sont en diagonale

pas au dessus d'un support adverse

— ko : j'ai pu passer en diagonale par dessus un support adverse

maximum 1 lancer par tour de jeu

— ko : la méthode `Game::passe(int dx, int dy)` ne tient pas à jour un compteur de passe : c'est géré dans le contrôleur

terminer tour de jeu au moins une action obligatoire

— ko : la méthode `Game::swapPlayer()` est publique et ne vérifie *rien*

possibilité de ne pas réaliser 3 actions

— ok dans classe métier mais ko dans contrôleur

fin de partie balle sur ligne adverse

ok, mais plantage (du contrôleur ?) : voir bogues non signalés

anti-jeu

— une méthode `Game::antiJeu()` est implémentée mais jamais invoquée, même pas dans contrôleur : les étudiant signalent dans le rapport qu'elle ne fonctionne pas bien

méthodes complètes : 1 méthode / 1 action de jeu

— la méthode `Game::isOver()` n'est utilisée par aucune méthode des classes métier

impossibilité de tricher (bibliothèque)

— la méthode `Game::swapPlayer()` est publique et ne vérifie *rien*
— la méthode `Game::move(int ox, int oy, int dx, int dy)` ne tient pas à jour un compteur de déplacement : ko, de plus il est possible de jouer alors que la partie est terminée
— la méthode `Game::passe(int dx, int dy)` ne tient pas à jour un compteur de passe : ko, de plus il est possible de jouer alors que la partie est terminée...

contrôleur

fiabilisation lectures clavier
- boucle (signalée) si pas entier fourni quand entier attendu

convivialité
- pas de choix de taille du plateau : pour masquer bogue (pas signalé dans rapport) du placement de balle si pas 7 x 7 ?
- indication du nom du joueur courant mais pas de sa position ni de sa couleur : il est impossible de savoir qui joue !
- il n'y a pas de légende sur les lignes / colonnes du plateau : je refuse de passer mon temps à compter et décide de ne pas tester l'application en détail mais de me fier à ce que je vois dans les sources pour ce qui concerne les déplacement, lancer et fin de partie. d'autant plus qu'il n'est pas indiqué s'il faut d'avord fournir ligne ou colonne... apparemment c'est ligne puis colonne et ça commence à l'indice 0
- je n'arrive pas à réaliser la moindre passe sur la ligne de départ malgré des coordonnées qui fonctionnent bien pour les déplacements. j'arrive à faire une passe vers la 2e ligne en diagonale
- convivialité très mauvaise

vue

design pattern observer
- la classe `Observable` est abstraite : *ko*
- pour le reste presque ok : `Game::move(int ox, int oy, int dx, int dy)` et `Game::passe(int dx, int dy)` notifient, ainsi que `Game::start()`, *mais pas* `Game::swapPlayer()`

absence de flux (cout) dans classes métier ko ! `cout` dans :
- `Game::start()`

autre (void)

gui

remise

retard (void)

autre (void)

documentation

ok

rapport

format pdf ok

bogue non signalé

— constat : je désire lancer la balle : au final 3 pions jaune ont une balle ! ensuite impossible de déplacer des pions qui pourraient le faire dans le respect des règles. . .

écart / ajout non signalé (void)

autre (void)

code source

portabilité

casse noms fichiers (void)

séparateur / (void)

c++ standard + qt (void)

si pas std + qt : portabilité (void)

gestion de la mémoire

— fuite mémoire dans `MainWindow::addPlayers()` : `Players * p = new Players(name,color)`

contrôleur

respect des règles
- — pas de configuration du plateau : taille
- — pas de configuration du plateau : variante
- — possible de terminer son tour alors qu'aucune action réalisée
- — possible de continuer à jouer alors que partie terminée

convivialité
- — les indications pour déplacer / lancer sont erronnées : il faut faire 2 clics sur plateau puis cliquer sur le bouton d'action
- — pas 100 % clics sur plateau : il faut indiquer l'action désirée via un bouton y compris pour bouger ou lancer la balle
- — obligation d'explicitement terminer son tour même si plus d'action disponible
- — système de désélection avec un bouton dédié : ok

vue

design pattern observer ok

convivialité
- — affichage pas très joli, mais ça fait le boulot
- — affichage des destinations possibles : ni déplacement, ni lancer
- — notification par messagebox qu'on n'a plus de déplacement ou de lancer : assez irritant

autre (void)

examen

(void)