Abalone : n7_45502_52646

nvs

31 mai 2021

Abalone : n7 (45502 & 52646)

dépôt

https://git.esi-bru.be/45502/dev4-abalone

`git@git.esi-bru.be:45502/dev4-abalone.git`

modélisation

remise

tag / commit ok : il y a un tag `modelization-release`

retard (void)

autre (void)

analyse

(void)

console

à partir de la remise console, l'étudiant 52646 ne fait plus partie du binôme.

remise

tag / commit ok : il y a un tag `console-release`

retard (void)

autre (void)

documentation

- je ne trouve aucun fichier de configuration pour doxygen
- la classe `View` n'est pas documentée
- le reste des classes et énumérations est documenté, mais les fichiers ne le sont pas, cela pourrait poser problème pour la génération de la documentation des énumérations `Color` et `State`

rapport

format pdf ok

bogue non signalé ko :

- j'ai pu déplacer 4 pions `@` qui ont poussé 1 `O` ou encore 4 `O` qui ont poussé 2 `@`

écart / ajout non signalé (void)

autre (void)

rapport / code

avertissement restant rem. : les subdirs ne fonctionnent pas sur ma machine linux. je n'ai pas envie de creuser. j'ai donc créé un projet avec tous les sources nécessaires. bon j'ai trouvé le souci. c'était un problème d'affichage dans qt creator... c'est pas grave, ça le fait aussi.

gcc signalé

(void)

non signalé

(void)

gcc + clang-analyzer

```
View.cpp:112:13: warning: Although the value stored to 'x' is used in the enclosing expression, the value is never actually read from 'x'
    return (x = x - 32);
            ^   ~~~~~~
1 warning generated.
```

faux positif

```
Board.cpp:121:13: warning: Although the value stored to 'i' is used in the enclosing expression, the value is never actually read from 'i
    return (i = m_configConfirmation.size()) ? true:false;
            ^   ~~~~~~~~~~~~~~~~~~~~~~~~~~
```

à régler

```
Board.cpp:174:13: warning: Branch condition evaluates to a garbage value [core.uninitialized.Branch]
        if (tempX && tempY){
            ^~~~
```

à régler

```
Board.cpp:174:22: warning: Branch condition evaluates to a garbage value [core.uninitialized.Branch]
        if (tempX && tempY){
                     ^~~~~
3 warnings generated.
```

à régler

```
Game.cpp:10:5: warning: Potential memory leak [cplusplus.NewDeleteLeaks]
    Player p1 = *new Player(WHITE,3);
    ^~~~~~~~~
Game.cpp:11:5: warning: Potential memory leak [cplusplus.NewDeleteLeaks]
```

```
    Player p2 = *new Player(BLACK,3);
    ^~~~~~~~
2 warnings generated.
```

à régler

clang++ (void)

clang++ + clang-analyzer idem g++

cppcheck signalé

(void)

non signalé

```
Board.h:76:10: performance:inconclusive: Technically the member function 'Board::diffMarble' can be static (but you may consider moving t
    bool diffMarble(const int, const int);
         ^
Board.cpp:162:13: note: Technically the member function 'Board::diffMarble' can be static (but you may consider moving to unnamed namespa
bool Board::diffMarble(int const diffX, int const diffY){
            ^
Board.h:76:10: note: Technically the member function 'Board::diffMarble' can be static (but you may consider moving to unnamed namespace)
    bool diffMarble(const int, const int);
         ^
```

à régler éventuellement

```
Board.h:95:10: performance:inconclusive: Technically the member function 'Board::isPlayableMarble' can be static (but you may consider mo
    bool isPlayableMarble(std::vector<Color>);
         ^
Board.cpp:263:13: note: Technically the member function 'Board::isPlayableMarble' can be static (but you may consider moving to unnamed n
bool Board::isPlayableMarble(std::vector<Color> vectorColor){
            ^
Board.h:95:10: note: Technically the member function 'Board::isPlayableMarble' can be static (but you may consider moving to unnamed name
    bool isPlayableMarble(std::vector<Color>);
         ^
```

à régler éventuellement

```
Board.cpp:163:9: style: The scope of the variable 'validNB' can be reduced. [variableScope]
    int validNB[3] = {-1, 0, 1};
        ^
```

à régler éventuellement

```
Board.cpp:164:10: style: The scope of the variable 'tempX' can be reduced. [variableScope]
    bool tempX, tempY;
         ^
```

à régler éventuellement

```
Board.cpp:164:17: style: The scope of the variable 'tempY' can be reduced. [variableScope]
    bool tempX, tempY;
                ^
```

à régler éventuellement

```
Board.cpp:118:13: style: Consider using std::count_if algorithm instead of a raw loop. [useStlAlgorithm]
            i++;
            ^
```

à régler éventuellement

```
Game.h:87:11: style:inconclusive: Technically the member function 'Game::getState' can be const. [functionConst]
    State getState();
          ^

Game.cpp:106:13: note: Technically the member function 'Game::getState' can be const.
State Game::getState(){
            ^

Game.h:87:11: note: Technically the member function 'Game::getState' can be const.
    State getState();
          ^
```

à régler

```
Game.h:92:25: style:inconclusive: Technically the member function 'Game::getPlayers' can be const. [functionConst]
    std::vector<Player> getPlayers();
```

```
                        ^
Game.cpp:110:27: note: Technically the member function 'Game::getPlayers' can be const.
std::vector<Player> Game::getPlayers(){
                          ^
Game.h:92:25: note: Technically the member function 'Game::getPlayers' can be const.
    std::vector<Player> getPlayers();
                        ^
```

à régler

```
Game.h:97:12: style:inconclusive: Technically the member function 'Game::getCurrent' can be const. [functionConst]
    Player getCurrent();
           ^
Game.cpp:114:14: note: Technically the member function 'Game::getCurrent' can be const.
Player Game::getCurrent(){
             ^
Game.h:97:12: note: Technically the member function 'Game::getCurrent' can be const.
    Player getCurrent();
           ^
```

à régler

```
Marble.cpp:13:29: style:inconclusive: Function 'setColor' argument 1 names different: declaration 'color' definition 'x'. [funcArgNamesDi
void Marble::setColor(Color x){
                            ^
Marble.h:26:25: note: Function 'setColor' argument 1 names different: declaration 'color' definition 'x'.
    void setColor(Color color);
                        ^
Marble.cpp:13:29: note: Function 'setColor' argument 1 names different: declaration 'color' definition 'x'.
void Marble::setColor(Color x){
                            ^
```

à régler éventuellement

```
Observer.h:32:17: performance:inconclusive: Technically the member function 'Observer::convCtoC' can be static (but you may consider movi
    std::string convCtoC(Color);
                ^
Observer.cpp:50:23: note: Technically the member function 'Observer::convCtoC' can be static (but you may consider moving to unnamed name
```

```
std::string Observer::convCtoC(Color c){
                      ^

Observer.h:32:17: note: Technically the member function 'Observer::convCtoC' can be static (but you may consider moving to unnamed namesp
    std::string convCtoC(Color);
                ^
```

à régler éventuellement

```
Player.h:32:9: style:inconclusive: Technically the member function 'Player::getDeadMarble' can be const. [functionConst]
    int getDeadMarble();
        ^

Player.cpp:8:13: note: Technically the member function 'Player::getDeadMarble' can be const.
int Player::getDeadMarble(){
            ^

Player.h:32:9: note: Technically the member function 'Player::getDeadMarble' can be const.
    int getDeadMarble();
        ^
```

à régler

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

Player.cpp:16:14: note: Technically the member function 'Player::isPlayerDead' can be const.
bool Player::isPlayerDead(){
             ^

Player.h:42:10: note: Technically the member function 'Player::isPlayerDead' can be const.
    bool isPlayerDead();
         ^
```

à régler

```
View.h:17:10: performance:inconclusive: Technically the member function 'View::title' can be static (but you may consider moving to unnam
    void title() const;
         ^

View.cpp:8:12: note: Technically the member function 'View::title' can be static (but you may consider moving to unnamed namespace).
void View::title() const{
           ^
```

```
View.h:17:10: note: Technically the member function 'View::title' can be static (but you may consider moving to unnamed namespace).
    void title() const;
         ^
```

à régler éventuellement

```
View.h:18:10: performance:inconclusive: Technically the member function 'View::displayWinner' can be static (but you may consider moving
    void displayWinner(std::string const& , std::string const &, Color) const;
         ^
View.cpp:23:12: note: Technically the member function 'View::displayWinner' can be static (but you may consider moving to unnamed namespa
void View::displayWinner(std::string const &t1, std::string const &t2, Color color) const{
           ^
View.h:18:10: note: Technically the member function 'View::displayWinner' can be static (but you may consider moving to unnamed namespace
    void displayWinner(std::string const& , std::string const &, Color) const;
         ^
```

à régler éventuellement

```
View.h:19:10: performance:inconclusive: Technically the member function 'View::displayStr' can be static (but you may consider moving to
    void displayStr(std::string const &);
         ^
View.cpp:27:12: note: Technically the member function 'View::displayStr' can be static (but you may consider moving to unnamed namespace)
void View::displayStr(std::string const &s){
           ^
View.h:19:10: note: Technically the member function 'View::displayStr' can be static (but you may consider moving to unnamed namespace).
    void displayStr(std::string const &);
         ^
```

à régler éventuellement

```
View.h:20:10: performance:inconclusive: Technically the member function 'View::displayStrC' can be static (but you may consider moving to
    void displayStrC(const std::string &, Color);
         ^
View.cpp:31:12: note: Technically the member function 'View::displayStrC' can be static (but you may consider moving to unnamed namespace
void View::displayStrC(std::string const &p, Color c){
           ^
View.h:20:10: note: Technically the member function 'View::displayStrC' can be static (but you may consider moving to unnamed namespace).
    void displayStrC(const std::string &, Color);
```

```
         ^
```

à régler éventuellement

```
View.h:24:10: performance:inconclusive: Technically the member function 'View::numberV' can be static (but you may consider moving to unn
    bool numberV(unsigned);
         ^
View.cpp:104:12: note: Technically the member function 'View::numberV' can be static (but you may consider moving to unnamed namespace).
bool View::numberV(unsigned number){
          ^
View.h:24:10: note: Technically the member function 'View::numberV' can be static (but you may consider moving to unnamed namespace).
    bool numberV(unsigned);
         ^
```

à régler éventuellement

```
View.h:25:10: performance:inconclusive: Technically the member function 'View::toUpperCase' can be static (but you may consider moving to
    char toUpperCase(char);
         ^
View.cpp:111:12: note: Technically the member function 'View::toUpperCase' can be static (but you may consider moving to unnamed namespac
char View::toUpperCase(char x){
          ^
View.h:25:10: note: Technically the member function 'View::toUpperCase' can be static (but you may consider moving to unnamed namespace).
    char toUpperCase(char);
         ^
```

à régler éventuellement

```
Game.cpp:96:0: style: The function 'getBoard' is never used. [unusedFunction]

^
Player.cpp:16:0: style: The function 'isPlayerDead' is never used. [unusedFunction]

^
```

ok

```
nofile:0:0: information: Cppcheck cannot find all the include files (use --check-config for details) [missingIncludeSystem]
```

ok

code source

portabilité

casse noms fichiers (void)

séparateur / (void)

c++ standard (void)

si pas std : portabilité (void)

bonnes pratiques

déclarations anticipées si possible
- `#include "Marble.h"` inutile dans `Board.h`
- `#include "Color.h"` manquant dans `Board.h`
- `#include <memory>` inutile dans `Game.h`

using namespace dans .h (void)

autre (void)

gestion de la mémoire
- on a `new` et `delete` dans des méthodes de `Board`, mais cette classe n'est pas équipée de destructeur, constructeurs de recopie, de déplacement, opérateurs d'assignation par recopie, par déplacement : pas ok même si pas de fuite mémoire avec `Board` dans l'application fournie
- on a une fuite mémoire dans `Game` : seul le 1er joueur est détruit. Q : pourquoi faire si compliqué pour créer les joueurs ?

classes métier

initialisation plateau et billes

- `Board::isSetUp()` ne retourne pas la valeur attendue logiquement
- `Board::diffMarble()` contient des variables non initialisées qui peuvent poser problème
- sinon plateau bien initialisé

joueurs (éventuellement)

- fuite mémoire avec les joueurs (voir + haut)

mouvement en ligne partie en cours

ok

sélection d'une bille de départ

ok dans `Board::slideOneMarble()` appelé par `Game::moveMarble()`

sélection d'une bille de départ du joueur actif

ok dans `Board::slideOneMarble()` appelé par `Game::moveMarble()`

maximum 3 billes du joueur actif en mouvement

ko :

- j'ai pu déplacer 4 pions `@` qui ont poussé 1 `O`

emplacement libre dernière dernière bille en mouvement

ok

règle de poussée des billes adverses ok

ko :

- si je tente une poussée impossible, le mouvement est refusé, mais c'est au joueur suivant de jouer
- pour le reste c'est ok

pas de suicide (éjection d'une de ses propres billes)

ko : il est possible de se suicider

déplacement effectif

ok

traitement de l'éjection d'une bille adverse

ok

détection de fin de partie

ok

alternance des joueurs

ok : bien pris en charge par `Game::moveMarble()`

mouvement de côté pas implémenté.

partie en cours

ko

sélection d'un ensemble de billes de départ

ko

sélection d'un ensemble de billes de départ du joueur actif

ko

sélection d'un ensemble de billes de départ alignées du joueur actif

ko

maximum 3 billes du joueur actif en mouvement

ko

emplacements libres dernière dernière les billes en mouvement

ko

pas de poussée de bille adverse

ko

pas de suicide (éjection d'une ou plusieurs de ses propres billes)

ko

déplacement effectif

ko

alternance des joueurs

ko

méthodes complètes : 1 méthode / 1 action de jeu

(void)

impossibilité de tricher (bibliothèque)

ok : `Game::moveMarble()` gère bien le jeu

contrôleur

fiabilisation lectures clavier ko :

— fournir `1`, `1`, `A`, `A` et ça plante

convivialité

— plateau hexagonal et coordonnée abapro : ok
— il n'y a pas de légende de couleur des pions : où sont les pions noirs, blancs ?
— plantage exception pas catchée si on ne sélectionne pas un pion de sa propre couleur

vue

design pattern observer l'implémentation de design pattern O / SdO à l'envers peut être ok, mais ce n'est pas le cas ici à mon avis. il faut que le SdO (Observable, Game) préviennent les Observer quand il se passe quelque chose. ici, le Game boucle dans `Observable::run()` mais si les observateurs ne font rien, on va avoir une boucle à vide inutile qui utilise des ressources pour rien : pas ok même si ici ça le fait car les observateurs sont aussi contrôleurs, mais en règle générale, un observateur n'est *pas* un contrôleur

absence de flux (cout) dans classes métier il y a des `cout` dans :

- `Board::clear()`
- `Game::clearPlayer()`

autre (void)

gui

remise

tag / commit ok : il y a un tag `gui-release`

retard (void)

autre (void)

documentation

- pas de fichier de configuration pour doxygen fourni
- les classes graphiques `MainWindow` et `HexCell` ne sont pas documentées

rapport

format pdf ok

bogue non signalé (void)

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
— fuites mémoire dans `Game::Game()`

contrôleur

respect des règles semble ok... sauf que pas de mouvement latéral

convivialité
— l'obligation de confirmer via un bouton dédié est un peu lourde, mais ça le fait

vue

design pattern observer ko : pas d'O / SdO dans l'application gui. voir application console où le contrôleur est l'observateur. ceci n'est pas possible avec application à interface graphique car on ne contrôle pas le flux des actions de l'utilisateur

convivialité ok : la vue est claire, on voit où on en est

autre (void)

examen

voir prise de notes manuscrites