Diaballik : n11_43256_51999

nvs

23 juin 2020

Diaballik : n11_43256_51999

console

remise

retard (void)

autre (void)

documentation

— documentation inutilisable sur « petit » écran : l'image dans le header de chaque page occupe les 2 / 3 de l'écran ! je renomme le fichier pour casser le lien vers l'image
— fichiers pas documentés avec `\file`
— pour le reste documentation *très complète*

rapport

format pdf (void)

bogue non signalé (void)

écart / ajout non signalé

— les étudiant ont considéré que dans le cas de la variante, l'objectif des pions R qui demarrent sur la ligne J adverse est la ligne de départ des autres pions R. ceci n'apparaît pas dans les règles du jeu !

autre (void)

rapport / code

avertissement restant

gcc signalé

(void)

non signalé

(void)

gcc + clang-analyzer (void)

clang++ (void)

clang++ + clang-analyzer (void)

cppcheck signalé

(void)

non signalé

(void)

code source

portabilité

casse noms fichiers (void)

séparateur / (void)

c++ standard (void)

si pas std : portabilité (void)

bonnes pratiques

déclarations anticipées si possible

- `#include "src/model/headers/Diaballik.hpp"` inutile dans `Controller.hpp` : déclaration anticipée suffit
- `#include "src/view/headers/View.hpp"` inutile dans `Controller.hpp` : déclaration anticipée suffit
- `#include "src/controller/headers/DiaballikAbstractEvent.hpp"` inutile dans `DiaballikEvent.hpp` : déclaration anticipée suffit
- `#include "src/controller/headers/DiaballikEvent.hpp"` inutile dans `EventFactory.hpp` : déclaration anticipée suffit
- `#include "src/view/headers/View.hpp"` inutile dans `EventFactory.hpp` : déclaration anticipée suffit : `class View`, `class Diaballik`
- `#include "src/view/headers/View.hpp"` inutile dans `HelpEvent.hpp` : déclaration anticipée suffit
- `#include "src/model/headers/Diaballik.hpp"` inutile dans `MoveEvent.hpp` : déclaration anticipée suffit
- `#include "src/view/headers/View.hpp"` inutile dans `MoveEvent.hpp` : déclaration anticipée suffit
- `#include "src/model/headers/Diaballik.hpp"` inutile dans `PassEvent.hpp` : déclaration anticipée suffit
- `#include "src/view/headers/View.hpp"` inutile dans `PassEvent.hpp` : déclaration anticipée suffit
- `#include "src/model/headers/Diaballik.hpp"` inutile dans `PassTurnEvent.hpp` : déclaration anticipée suffit
- `#include "src/model/headers/Diaballik.hpp"` inutile dans `SelectEvent.hpp` : déclaration anticipée suffit
- `#include "src/view/headers/View.hpp"` inutile dans `SelectEvent.hpp` : déclaration anticipée suffit
- `#include "src/view/headers/View.hpp"` inutile dans `ShowEvent.hpp` : déclaration anticipée suffit : `class View`, `class Diaballik`
- `#include "Position.hpp"` inutile dans `Board.hpp` : déclaration anticipée suffit
- `#include "src/view/headers/Observer.hpp"` inutile dans `Observable.hpp` : déclaration anticipée suffit
- `#include "src/model/headers/Diaballik.hpp"` inutile dans `View.hpp` : déclaration anticipée suffit : `class Diaballik`, `class Piece`

using namespace dans .h ok

autre

- inclusion d'un fichier d'en-têtes C plutôt que C++ dans `Configs.hpp`
- inclusion inutile de `iostream` dans `ViewConsole.hpp`
- inclusion inutile de `iostream` dans `View.hpp`
- inclusion manquante de `iostream` dans `ViewConsole.cpp`

gestion de la mémoire

- dans `DiaballikEventFactory::generateEvent(std::string input)`, multitude (8) de `DiaballikEvent { new XxxEvent { ... } }`. on a bien un destructeur dans la classe `DiaballikEvent`, mais pas de constructeur par recopie / déplacement ni opérateur d'assignation par recopie / déplacement. pas de fuite mémoire observée dans valgrind, mais danger.

classes métier

initialisation plateau

- pas de limitation de la taille du plateau à 5, 7 ou 9

supports

ok

balles

ok

possibilité de variante

- oui mais les étudiant ont considéré que dans le cas de la variante, l'objectif des pions R qui demarrent sur la ligne J adverse est la ligne de départ des autres pions R. ceci n'apparaît pas dans les règles du jeu !

joueurs (éventuellement)

(void)

déplacement support latéral uniquement

- ko : il est possible de passer en diagonale un chemin latéralement bloqué par des supports

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

ok

lancer balle latéral ou diagonal

ok

pas au dessus d'un support adverse

ok

maximum 1 lancer par tour de jeu

ok

terminer tour de jeu au moins une action obligatoire

— ko : la méthode `Diaballik::passTurn()` ne vérifie pas si des actions ont été faites lors du tour du joueur

possibilité de ne pas réaliser 3 actions

ok

fin de partie balle sur ligne adverse

ok

anti-jeu

— il existe une méthode `Diaballik::checksAntiGame()` : je ne l'ai pas testée

méthodes complètes : 1 méthode / 1 action de jeu

- la méthode `Diaballik::isOver()` n'est appelée par aucune méthode de `Diaballik` : la fin de partie n'est donc jamais testée dans les classes métier...

impossibilité de tricher (bibliothèque)

- la méthode `Diaballik::select(const Position & pos)` ne vérifie pas si partie terminée : ce n'est pas nécessairement un soucis si ensuite c'est bien pris en charge par les métodes de déplacement de support ou de lancer de balle, ce qui n'est pas le cas ici !
- la méthode `Diaballik::movePiece(const Position & pos)` ne vérifie pas si le jeu est terminé : il est donc possible de continuer de jouer alors que la partie est finie !
- la méthode `Diaballik::throwBall(const Position & pos)` ne vérifie pas si le jeu est terminé : il est donc possible de continuer de jouer alors que la partie est finie !
- la méthode `Diaballik::passTurn()` est publique et ne vérifie pas si des actions ont été faites lors du tour du joueur

contrôleur

fiabilisation lectures clavier ok

convivialité

- dans le cas de la variante, pas de marque des pions d'objectif différent tel que considéré (à tort) par les étudiants
- marquage de la pièce sélectionnée
- indication de destinations possibles suite à la sélection d'un support
- obligation de donner l'ordre pour terminer son tour, même si plus d'action disponible

vue

design pattern observer ok

absence de flux (cout) dans classes métier (void)

autre (void)

gui

remise

retard (void)

autre (void)

documentation

ok

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 (void)

contrôleur

respect des règles
- possible de terminer son tour alors qu'aucune action réalisée

convivialité
- obligation d'explicitement terminer son tour même si plus d'action disponible
- facile et presque intuitif (clic droit pour déselectionner support avec balle, pas d'explications en fait si dans une boîte de dialogue mais je ne l'ai pas toujours vue) : convivial

vue

design pattern observer ok

convivialité
- affichage des destinations possibles : déplacement et lancer
- ça déborde un petit peu dans le cadre à droite : convivial ok
- en fin de partie : le plateau n'est plus affiché mais message de victoire : ok mais laisser le plateau serait mieux à mon avis

autre
- rappel : ces étudiants considèrent que dans la variante l'objectif des pions qui commencent sur la ligne adverse est leur propre ligne

examen

(void)