Université Claude Bernard Lyon 1 Département Informatique - 3A
Polytech

Projet Tutoré

Projet 2048

Critères d'évaluation :

- Qualité de l'analyse et du code associé
- Respect du Modèle Vue Contrôleur Strict
- Extensions proposées

Travail à réaliser obligatoirement en binôme :

- Travail personnel entre les séances
- Évaluations individuelles lors de la démonstration (dernière séance)
- Rapport PDF de 3 à 6 pages (et code source) à déposer sur Tomuss 1 semaine après la démonstration : liste des fonctionnalités (et contributions par étudiant), 1 diagramme UML au choix, justification de l'analyse, copies d'écran

1 2048

Soit une grille de taille variable. Les cases associées à une valeur sont déplacées à chaque étape dans la direction indiquée par le joueur (haut, bas, gauche, droite), les cases ainsi rassemblées et de valeurs identiques sont alors fusionnées avec une valeur doublée. À chaque étape, une case de valeur 2 ou 4 apparaît aléatoirement. L'objectif est d'obtenir une case de valeur 2048. Lorsqu' aucun mouvement ne peut être réalisé, sans case de valeur 2048 ou plus, la position est qualifiée de perdante. Voir une démonstration ici : `https://jeu2048.fr`

FIGURE 1 – une position de jeu, image extraite de la description wikipedia du 2048 : `https://fr.wikipedia.org/wiki/2048_(jeu_vidéo)`

2 Travail à réaliser

2.1 Comprendre le code fourni

— l'application affiche aléatoirement l'état d'une case (vide ou valeur numérique), sur une vue Console ou Graphique.
— Identifier les parties Modèle, Vue et Contrôleur dans le code fourni.
— La partie Modèle (processus métier, données métier) n'est pas développée, lorsque vous développez le modèle, veillez à ce que les vues proposées fonctionnent toujours.

FIGURE 2 – Vue Swing (gauche) et vue Console (droite)

2.2 Développer l'application

L'application doit être développée en suivant une démarche objet : chaque objet Case se déplace et interagit avec son environnement.

— Le contrôleur n'invoque plus la fonction rnd() de Jeu, mais une fonction *action(Direction)* qui (1) déclenche de façon appropriée le déplacement des cases (fonction *déplacer(Direction)* des cases), et (2) ajoute aléatoirement les nouvelles cases tout en détectant les positions perdantes.
— la fonction *déplacer(Direction)* de la classe Case permet à une case de de percevoir son voisinage, se déplacer et fusionner (en cas de comptabilité). A vous de voir comment mettre en place la fusion de cases.

La qualité du code est évaluée, le traitement doit être réparti entre les classes Jeu et Case (ne pas écrire un algorithme central qui utilise la classe case comme une simple structure de données).

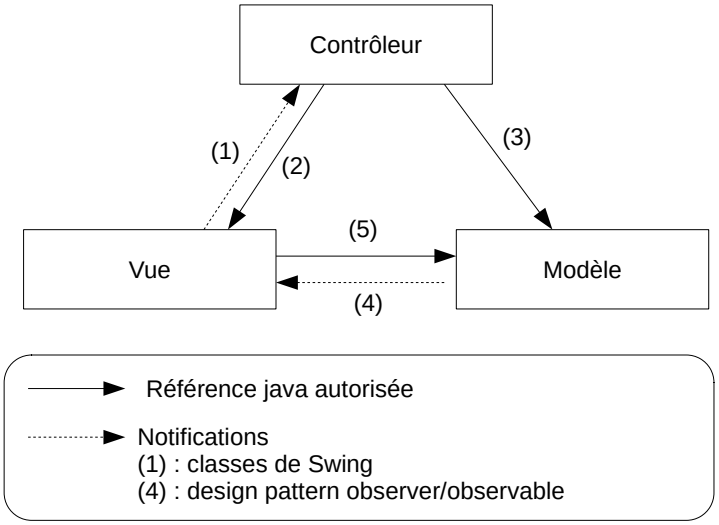
2.3 Développer une extension ou plus

Exemples (difficulté croissante) :

- améliorer le rendu de la vue graphique : coloration des cases, etc.
- utiliser un pool de processus (classe Executor) au lieu des fonctionnalités directes de la classe Thread
- enregistrer les données utilisateurs (meilleurs score, temps, etc.)
- une possibilité de déblocage de votre choix (bonus) proposée au joueur
- console avancée et interface graphique étendue : menus de navigation
- possibilité de reprendre des coups (historique)
- une version deux joueurs (deux grilles côte à côte, les joueurs jouent alternativement, les coups sont appliqués aux deux grilles, le premier joueur dans une position bloquée perd la partie)
- animations : déplacement des tuiles case par case, avec temps de pause
- développer une version octogonale du jeu (pour cela, dessiner vous-même le composant graphique associé à la grille, conserver la compatibilité avec la version standard en gardant le code générique). Autres variantes : `https://jeu2048.fr/`
- développer une IA (pour chaque direction jouer des parties par tirages aléatoires, estimer le meilleur coup, développer un MCTS si vous le souhaitez (compliqué et très long) `https://fr.wikipedia.org/wiki/Recherche_arborescente_Monte-Carlo`). L'IA peut alors être exploitée de différentes façons (définir la difficulté des cases ajoutées, estimer la résolubilité de la position, signaler les erreurs commises, etc.)

Vous pouvez également proposer des extensions.

3 Rappel Modélisation MVC Strict



MVC Strict :
(1) Récupération de l'événement Swing par le controleur
(2) Répercutions locale directe sur la vue sans exploitation du modèle
(3) Déclenchement d'un traitement pour le modèle
(4) Notification du modèle pour déclencher une mise à jour graphique
(5) Consultation des données pour réaliser la mise à jour Graphique

Application Calculette :
(1) récupération clics sur bouton de calculette
(2) construction de l'expression dans la vue (« 1+4+ »)
(3) déclenchement calcul (suite à (1) bouton « = »)
(4) Calcul terminé, notification de la vue
(5) La vue consulte le résultat et l'affiche