Matricule : 54309

Matricule : 54383

Rapport d’analyse Baba Is You

"Baba Is You" est un jeu de puzzle créatif développé par Hempuli Oy.

Le jeu est construit de règles modifiables par le joueur pour atteindre l'objectif du niveau. Le joueur contrôle un personnage nommé "Baba" et doit résoudre des puzzles en modifiant les règles du jeu. Les puzzles deviennent de plus en plus complexes à mesure que le niveau de jeu augmente.

Notre implémentation est faite avec le c++ 20 et les librairies qt creator utilisées pour la partie graphique.

Classes et méthodes de notre implémentation:

> ***Classe Game*** : La classe Game gère le fonctionnement de tous les objets du plateau de jeu. Elle initialise les autres éléments du jeu tels que le joueur, les niveaux et les règles. c’est l’observé de notre implémentation.
>
> - Méthodes :
>   - attach() et detach() : Permettent respectivement d’abonner/désabonner le modèle à la vue. Sans ces méthodes la classe Game ne pourra pas informer la vue sur les nouvelles modifications du jeu.
>   - receiveDirection() : Permet d'effectuer des traitements sur le plateau de jeu une fois que l’utilisateur entre une direction pour déplacer un objet.
>   - checkPosition() : Vérifie la validité d’une position par rapport aux limites du plateau de jeu.
>   - changePlayer() : Remplace le joueur courant par un objet donné, ceci dépend de la règle de connecteur “IS” et d’effet “YOU”. Exemple : On peut créer la règle “WALL IS YOU” dans ce cas l’objet “WALL” devient le “joueur” l'élément qui peut se déplacer.
>   - nextLevel() : Permet de passer au niveau suivant et donc de charger un nouveau fichier pour modifier le plateau de jeu.

Classe BoardView : La classe BoardView représente l'observateur(la vue) de notre implémentation, elle est responsable de l'affichage du plateau de jeu après une mise à jour du modèle.

- Méthodes :
  - update() : Gère la nouvelle version du plateau de jeu après une mise à jour du modèle.
  - print() : Affiche la nouvelle version du plateau de jeu.
  - askPosition() : Demande une position à l'utilisateur pour effectuer un déplacement sur le plateau de jeu.
  - askDirection() : Demande une direction à l'utilisateur pour effectuer un déplacement sur le plateau de jeu.
  - isInside() : Vérifie si une position est valide par rapport aux dimensions du plateau de jeu.

Classe ISubject : Classe de base des observés dont la classe Game utile pour le design-pattern observateur observé.

Classe IObserver : Classe de base des observateurs dont la classe BoardView utile pour le design-pattern observateur observé.

Classe ObjectManager : Gère toutes les fonctionnalités des objets du plateau de jeu.

- Méthodes :
  - push() : Cette méthode pousse un objet sur le plateau de jeu.
  - move() : Cette méthode déplace un objet sur le plateau de jeu.
  - canBePush : Cette méthode vérifie si un objet peut être poussé.
  - findAllMaterial() : Renvoie la position de tous les objets dont l'élément est donné, sera utile pour le changement d'état de ces objets.
  - updateAllState() : Change l'état de tous objets dont les positions sont données.
  - isValidRule() : Vérifie si une règle à une position donnée est valide.
  - addTarget() : Cette méthodes ajoute une nouvelle position d'une nouvelle cible, ceci est utile en fonction de l'élément de la règle de connecteur "IS", et d'effet "WIN". Exemple : "ROCK IS WIN" signifie que le joueur gagne la partie sur il atteint un "ROCK"

Classe FileManager : La classe FileManager représente la gestion des niveaux de jeu qui consiste à charger des fichiers pour mettre à jour le plateau de jeu.

- Méthodes :
  - save() : Cette méthode sauvegarde le niveau actuel dans un fichier donnée en paramètre.

- load() : Cette méthode charge un nouveau niveau de jeu donné par le fichier en paramètre.
- initBoard() : Initialise le plateau de jeu en chargeant le fichier donné en paramètre.
- setLevel() : Met à jour le niveau courant.
- changeLevel() : Change le niveau courant en mettant à jour le plateau de jeu.

Classe Player : La classe Player représente le personnage jouable tel que "Baba". Un joueur peut être également un des matériaux du plateau de jeu.

- Méthodes :
  - move()To : Cette méthode déplace le personnage à une position spécifique sur la plateau.
  - canMove() : Vérifie si le joueur actuel peut être déplacé, celà dépend de l'état de la règle de connecteur “IS” et d'effet “YOU”. Exemple : Si la règle “BABA IS YOU” est modifiée en “WALL IS YOU” alors le personnage “BABA” ne peut plus être déplacé.
  - setMovable() : Change l'état du joueur du point de vue du déplacement. Exemple : Si la règle “WALL IS YOU” est modifiée en “BABA IS YOU” le personnage “BABA” devient l'objet qui peut se déplacer sur le plateau.
  - pushObject() : Cette méthode permet au joueur de pousser un objet sur le plateau dans une direction donnée.

Classe Rule : La classe Rule représente une règle du jeu. Les règles sont des phrases telles que "BABA IS YOU" ou "WALL IS STOP".

- Méthodes :
  - isValid() : Cette méthode vérifie la validité d'une règle dont l'élément de la règle se trouve à la position donnée sur le plateau de jeu.

  Remarque : Il est possible qu'il manque des méthodes à cette classe, nous veillerons à les rajouter dans la partie console si nécessaires.

Classe ObjectState : Cette classe représente les états possibles des objets du plateau de jeu à savoir, pousser un matériel, traverser un matériel.

- Méthodes :
  - setPushable() : Cette méthode met à jour le fait qu'un objet puisse être poussé.
  - setTraversable() : Cette méthode met à jour le fait qu'un objet puisse être traversé.

Classe GameObject : Cette classe est la classe mère de tous les objets du jeu, y compris les matériaux et le joueur courant.

- Méthodes :
  - updatePushable() : Cette méthode met à jour le fait qu'un objet puisse être poussé.

Classe Material : Cette classe représente un matériel du jeu tel que : ROCK, GRASS, WALL etc.

Classe RuleElement : Cette classe représente un élément d'une règle et sa position sur le plateau de jeu, sera utile pour vérifier qu'une règle est valide.

Classe RuleConnector : Cette classe représente un connecteur d'une règle et sa position sur le plateau de jeu, sera utile pour vérifier qu'une règle est valide.

Classe RuleEffect : Cette classe représente un effet d'une règle et sa position sur le plateau de jeu, sera utile pour vérifier qu'une règle est valide.

Le jeu "Baba Is You" est un excellent exemple de programmation orientée objet. Les classes représentent des objets du monde réel tels que des personnages, des règles et des niveaux, et les méthodes définissent leur comportement. La structure de programmation du jeu permet aux joueurs de modifier les règles du jeu pour atteindre l'objectif du niveau, ce qui rend le jeu très créatif.