RAPPORT DE PROJET

Jeu de Mines

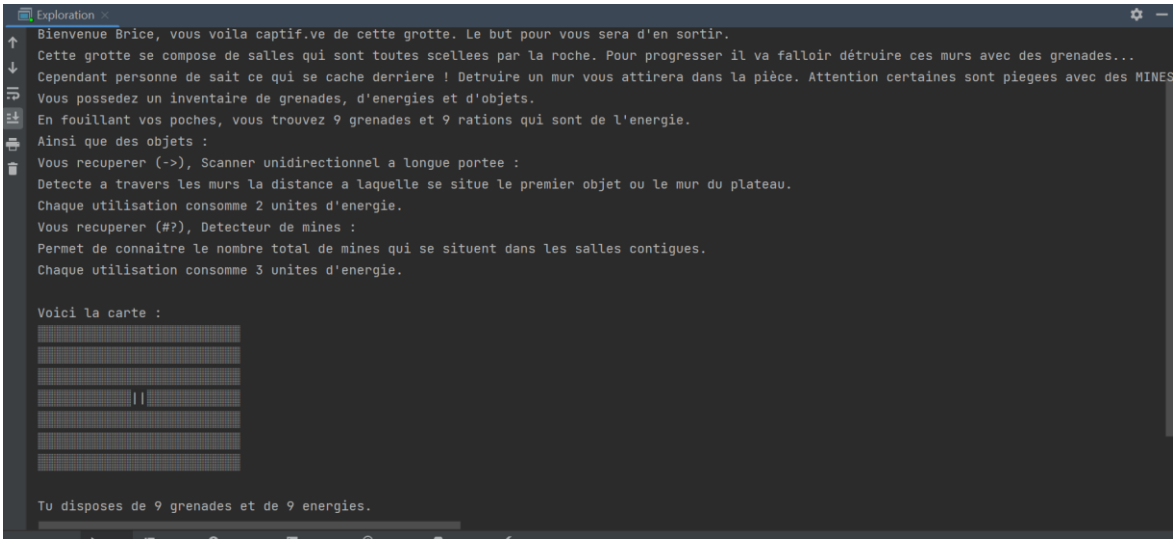
Brice LAWSON

Et

Thomas Moraux

Avril 2024

Table des Matières



Introduction

Le présent rapport détaille l'implémentation d'un jeu d'aventure textuel réalisé en Java. Ce jeu simule l'exploration d'un plateau composé de salles par un joueur, qui doit utiliser des ressources telles que des grenades et de l'énergie pour progresser et atteindre la sortie. Le jeu propose un environnement interactif avec des éléments tels que des mines. Notre approche se base sur la programmation orientée objet en Java pour concevoir et développer ce jeu. Ce document détaillera les règles du jeu, les paramètres à considérer, les recommandations de modélisation et les étapes de développement du programme.

Développement

Dans cette section, nous examinerons en détail les classes nouvellement introduites dans le code, en mettant l'accent sur leurs fonctionnalités, leurs interactions et leurs relations d'héritage.

2.1. Analyse des classes

ReserveEnergie

La classe **ReserveEnergie** représente une réserve d'énergie dans le jeu. Voici une analyse détaillée de cette classe :

- **Description** : La classe **ReserveEnergie** permet de gérer une réserve d'énergie dans le jeu. Elle offre la possibilité aux joueurs d'interagir avec cette réserve pour obtenir de l'énergie supplémentaire.
- **Attributs** :
    - **disponible** : Indique la quantité d'énergie disponible dans la réserve.
    - **ramasser** : Un booléen indiquant si la réserve a été ramassée par le joueur.
- **Méthodes importantes** :
    - **interaction(Joueur j)** : Permet au joueur d'interagir avec la réserve d'énergie. Si de l'énergie est disponible, le joueur peut en récupérer une certaine quantité.
    - **toString()** : Renvoie une représentation sous forme de chaîne de caractères de la réserve d'énergie.

Salle

La classe **Salle** représente une salle dans le jeu. Voici une analyse de cette classe :

- **Description** : La classe **Salle** modélise une salle du jeu. Chaque salle peut contenir un objet et avoir des accès à d'autres salles.
- **Attributs** :

- **visible** : Un booléen indiquant si la salle est visible ou non.
- **position** : La position de la salle sur le plateau.
- **plateau** : Le plateau auquel la salle appartient.
- **objet** : L'objet contenu dans la salle, s'il y en a un.
- **acces** : La liste des accès disponibles à partir de cette salle.

- **Méthodes importantes** :
  - **entree(Joueur j)** : Permet au joueur d'entrer dans la salle et d'interagir avec l'objet s'il y en a un.
  - **getVoisine(Direction d)** : Renvoie la salle voisine dans la direction donnée.

ScannerUnidirectionnel

La classe **ScannerUnidirectionnel** représente un outil de scan dans le jeu. Voici une analyse de cette classe :

- **Description** : Cette classe représente un scanner unidirectionnel à longue portée utilisé par les joueurs pour détecter des objets à travers les murs.
- **Attributs** :
  - **directionCourante** : La direction dans laquelle le scanner est orienté.
  - **ramasser** : Un booléen indiquant si l'outil a été ramassé par le joueur.
- **Méthodes importantes** :
  - **interaction(Joueur j)** : Permet au joueur d'interagir avec l'outil et de le ramasser.
  - **activation(Joueur j)** : Active le scanner dans la direction donnée et affiche la distance jusqu'à la prochaine salle avec un objet.

CaisseGrenades

La classe **CaisseGrenades** représente une caisse de grenades dans le jeu. Voici une analyse de cette classe :

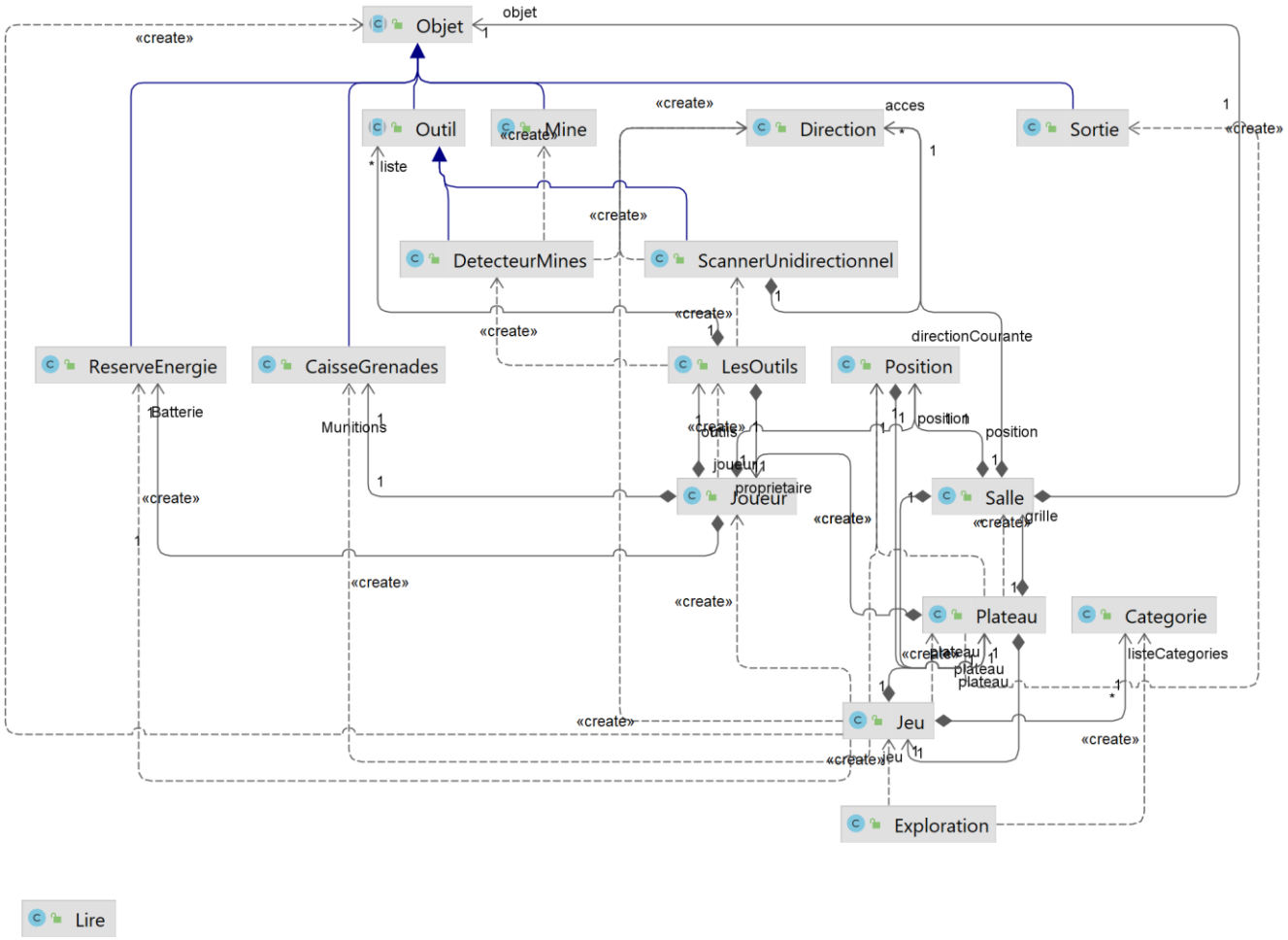
- **Description** : Cette classe gère une réserve de grenades disponibles pour les joueurs.
- **Attributs** :
  - **disponible** : La quantité de grenades disponibles dans la caisse.
  - **ramasser** : Un booléen indiquant si la caisse a été ramassée par le joueur.
- **Méthodes importantes** :
  - **interaction(Joueur j)** : Permet au joueur d'interagir avec la caisse pour obtenir des grenades.

- **toString()** : Renvoie une représentation sous forme de chaîne de caractères de la caisse de grenades.

2.2. Relations d'héritage

Les relations d'héritage entre les classes sont utilisées pour partager des fonctionnalités communes et établir une structure cohérente du code. Voici les principales relations d'héritage dans le code fourni :

- La classe abstraite **Objet** est la classe de base pour les objets du jeu. Les classes **Outil**, **ReserveEnergie**, **Sortie**, **Caissegreande**, **ScannerUnidirectionnel** et **CaisseGrenades** héritent de cette classe, partageant ainsi ses attributs et méthodes communs.
- La classe **Outil** hérite de la classe **Objet** et sert de classe de base pour les outils du jeu. La classe **ScannerUnidirectionnel** est une sous-classe de **Outil**, lui permettant de bénéficier des fonctionnalités communes aux outils. Pareil pour la classe **Detecteurmine**



Discussion

Dans cette section, nous discuterons des résultats et des observations concernant les classes nouvellement introduites, en analysant leurs points forts et faibles, en les comparant éventuellement à d'autres travaux similaires, et en proposant des suggestions pour des améliorations futures.

Interprétation des résultats et observations

Les classes introduites enrichissent significativement le jeu en offrant de nouvelles fonctionnalités et en élargissant les possibilités d'interaction pour les joueurs. La classe **ReserveEnergie** permet aux joueurs de récupérer de l'énergie supplémentaire, ce qui peut être crucial pour leur progression dans le jeu. De même, la classe **CaisseGrenades** offre la possibilité de récupérer des grenades, ce qui peut être essentiel pour combattre les ennemis ou surmonter des obstacles.

L'introduction de la classe **ScannerUnidirectionnel** ajoute une dimension stratégique au jeu en permettant aux joueurs de détecter des objets à travers les murs. Cela peut les aider à planifier leurs déplacements et à éviter les pièges potentiels. Enfin, la classe **Salle** fournit un cadre pour la modélisation des environnements du jeu, avec la possibilité de contenir des objets et d'avoir des accès à d'autres salles.

Analyse critique des points forts et faibles

Les points forts de ces classes incluent leur modularité et leur extensibilité. Elles sont conçues de manière à être facilement intégrées dans le jeu existant et à permettre l'ajout de nouvelles fonctionnalités à l'avenir. De plus, elles offrent une bonne encapsulation des données et des fonctionnalités, ce qui facilite la maintenance et le débogage du code.

Cependant, certains points faibles peuvent être identifiés. Par exemple, bien que les classes soient bien conçues individuellement, leur interaction et leur intégration dans le jeu peuvent nécessiter une attention particulière pour assurer une expérience de jeu fluide et cohérente. De plus, la gestion des erreurs et des exceptions peut être améliorée pour rendre le jeu plus robuste et plus convivial pour les utilisateurs.

Comparaison avec d'autres études ou travaux similaires

Bien qu'il existe de nombreux jeux avec des mécanismes similaires, chaque implémentation est unique et peut varier en fonction des objectifs du jeu et des préférences des développeurs. Dans certains jeux, par exemple, les réserves d'énergie et les caisses de munitions peuvent être combinées en un seul objet, tandis que dans d'autres, des outils de détection peuvent avoir des fonctionnalités plus avancées.

Cependant, les classes introduites ici s'inspirent des concepts couramment utilisés dans les jeux d'aventure et de stratégie, ce qui les rend familiers aux joueurs et aux développeurs. Leur conception modulaire et leur intégration dans le jeu peuvent être comparées à des approches similaires dans d'autres jeux, mettant en évidence les meilleures pratiques et les leçons apprises.

Suggestions pour des améliorations ou des pistes de recherche futures

Pour améliorer ces classes et enrichir davantage le jeu, plusieurs pistes de recherche peuvent être envisagées. Tout d'abord, l'ajout de mécanismes de génération procédurale pour les salles et les objets peut rendre le jeu plus dynamique et offrir une jouabilité accrue. De plus, l'introduction de mécanismes de combat plus élaborés, tels que des armes spéciales ou des capacités uniques, peut ajouter de la profondeur au gameplay.

En outre, l'optimisation des performances et la gestion efficace des ressources sont des aspects importants à prendre en compte, surtout si le jeu est destiné à être exécuté sur des plateformes avec des contraintes matérielles. Enfin, l'intégration de mécanismes multijoueur ou l’ ajout d’ennemis peuvent étendre les possibilités de jeu et offrir une expérience plus immersive.

En conclusion, les classes introduites dans cette discussion représentent une étape importante dans le développement du jeu, mais il reste encore beaucoup de potentiel à explorer pour créer une expérience de jeu captivante et innovante.