La chasse au trésor

AVENIA Adrien MARCHE Claire (groupe TP2A)

2017 – 2018

Introduction

Le sujet de ce projet était de créer un jeu de chasse au trésor dans lequel un trésor était placé au hasard sur une carte, et des chasseurs de trésor avaient pour objectif de l'atteindre, l'objectif du joueur étant alors de les en empêcher en posant des pierres, au nombre limité, en travers de leur chemin. Nous avons en plus ajouté d'autres types de chasseurs que le chasseur basique, qui sont l'escroc, qui a une probabilité de fausser les distances au trésor des cases qui l'entourent à chaque tour pour les autres chasseurs, en recevant lui-même les distances exactes depuis ces cases, et le chasseur furtif, qui a, à chaque tour où il est visible, une chance non nulle de devenir invisible pour un unique tour le tour suivant, obligeant le joueur à garder sa position en tête.

Nous avons fait le choix de prêter une importance particulière à l'ordre dans lequel les pierres, en nombre limité, sont posées. En effet, une fois que le joueur a posé une pierre, il ne peut pas faire lui-même le choix de la déplacer. Elle ne sera déplacée que lorsque le joueur n'aura plus de pierre en réserve et qu'elle sera la pierre placée la moins récemment. La victoire du joueur, dont la condition était laissée libre, se réalise lorsqu'aucun chasseur ne peut bouger, ou lorsque le trésor est entièrement entouré de pierres

1 Choix des structures de données

La grille sur laquelle se déroule le jeu consiste en une `ArrayList` de `Case`. Une méthode dans la classe `Grille` permet d'accéder à une case selon ses coordonnées. Ainsi, la création de la grille et l'insertion de cases en son intérieur est simplifiée.

Les pierres sont stockées dans une `ArrayList` contenue dans `Joueur` afin de pouvoir les stocker dans l'ordre de leur apparition, qui est nécessaire à leur déplacement quand le joueur n'a plus de pierres en réserve.

Les chasseurs sont stockés dans une `PriorityQueue` comme précisé dans le sujet. Ainsi, ils peuvent être ordonnées selon leur distance au trésor dès leur insertion. De plus, comme il s'agit d'une `PriorityQueue` de `Chasseur`, on peut y stocker les types de chasseurs particuliers dérivés de la classe `Chasseur` que sont l'`Escroc` et le `Furtif`. Lors de leur déplacement, pour être sûr de ne pas déplacer plusieurs fois le même chasseur, on vide la `PriorityQueue` à mesure des déplacements de ses membres, et on en remplit au même moment une `ArrayList` locale de `Chasseur`, qu'on réinsère ensuite un par un dans la `PriorityQueue` une fois le mouvement de tous les chasseurs terminé.

2 Bilan

Le projet tel qu'il a été réalisé respecte les règles élémentaires énoncées dans le sujet. Tous les éléments principaux qui y étaient décrits sont implémentés et fonctionnent selon la spécification qui en a été faite. Une condition de victoire et de défaite sont implémentées et toutes les deux fonctionnelles. De plus, deux

types de chasseurs spéciaux ont été ajoutés au jeu : l'escroc et le chasseur furtif, dont les déplacements et actions spéciales sont fonctionnelles. Enfin, à la fin de la partie, le joueur obtient un score dépendant du nombre de tours pendant lesquels il a réussi à protéger le trésor, augmenté, s'il a gagné, d'un nombre constant.

2.1 Bilan technique

Pour ce projet, nous avons utilisé la bibliothèque graphique Java Swing. Nous avons codé à la fois sur une IDE (Eclipse) et à l'aide d'un éditeur de texte (Notepad++) avec compilation en console.

2.2 Améliorations possibles

Il serait tout d'abord possible d'améliorer l'interface graphique minimaliste du programme, par exemple en utilisant des images pour représenter les différentes entités présentes sur la map.

On pourrait également imaginer des obstacles indépendants des actions du joueur et inamovibles, générés aléatoirement de telle manière que le trésor ne soit jamais isolé, afin de faciliter le rôle du joueur et de compliquer celui des chasseurs.

Il serait également possible d'employer un véritable algorithme de pathfinding pour les chasseurs, afin qu'ils soient moins facilement bloqués par un mur de pierres.

2.3 Bilan du travail en binôme

L'organisation du travail en binôme s'est réalisée rapidement. Nous avons commencé par la réalisation en commun de l'analyse et du diagramme de classe initial, puis la base du jeu a également été codée en commun, de telle sorte que le jeu puisse être fonctionnel, mais sans interface graphique. L'interface graphique et les différentes améliorations (dont l'affichage de la distance au trésor pour la démonstration) ont ensuite été réalisées séparément.