

# **Rapport de projet Programmation Objets Avancée**



**UNIVERSITÉ DE  
FRANCHE-COMTÉ**

## **Jeu de chasse au trésor**

**UFR Sciences et Techniques  
Licence informatique**

Par : Haouche Achour

Année universitaire:2019/2020

# Sommaire

Présentation de l'application

Choix de programmation

Class

Etats

Difficulté rencontrer

Optimisation à apporter

Description du jeu obtenu

Exemple du rendu de la partie console

Exemple du rendu de la partie graphique

Conception de l'application

Diagramme de classe

Bilan

Conclusion

## Présentation de l'application

**Le but du jeu** est de faire se déplacer plusieurs personnages (des aventuriers) sur un damier, tous à la recherche d'un trésor qui se trouve dans une case du damier inconnue des personnages. Le damier comprend également des murs qui feront obstacle aux déplacements des personnages. Chaque personnage se déplace tout seul, c'est à dire sans intervention de l'utilisateur. Son parcours est conditionné par les cases qu'il rencontre. Les personnages se déplacent d'une case à la fois dans une des huit directions possibles autour de sa case.

Les huit directions possibles du personnage P  
Lorsqu'un personnage veut se déplacer sur une case, la case visée peut être :  
- libre auquel cas le personnage peut l'occuper ;  
- déjà occupée par un autre personnage auquel cas le personnage devra changer de direction ;  
- une pierre faisant partie d'un mur que le personnage devra contourner ;  
- un bord du damier auquel cas le personnage devra changer de direction.  
Déplacement d'un personnage : Quand un personnage se déplace, il vise la case cible selon sa propre direction.

Le personnage P de direction 1 vise la case grisée (case cible). La case cible « réagit » au déplacement prévu par le personnage, selon son type. Réaction de la case cible selon son type :

1. Type Libre : le personnage peut l'occuper.

2. Type Libre mais occupée par un autre personnage :

le personnage ne bouge pas, il est réorienté au hasard dans une des huit directions possibles afin de l'éloigner du trésor.

3. Type Pierre (faisant partie d'un mur) : le personnage ne bouge pas, il est réorienté dans une direction (vers un des deux bouts du mur) qui lui permette à la fois de contourner le mur et aussi de se rapprocher le plus du trésor.  
4. Type Bord du damier : le personnage ne bouge pas, il est réorienté dans la direction symétrique à la sienne. Par exemple, la direction symétrique à la direction 1 est la direction

5. Type Trésor : le personnage a gagné. C'est la fin de la partie. Autrement dit, faire évoluer le jeu d'un tour consistera à déplacer chaque personnage un par un, en gérant leurs déplacements selon la case cible visée (ou case cible) par chaque personnage. La partie se termine dès qu'un personnage a comme case cible : la case Trésor.  
Les éléments du jeu : Le damier de dimension  $dim \times dim$  cases (de type Libre, Pierre ou Trésor), et aussi les cases pour les bords. Les murs sont horizontaux ou verticaux, constitués de cases de type Pierre. Pour ne pas bloquer un personnage, les murs ne seront jamais collés les uns aux autres. De plus, ils ne sont jamais collés non plus aux bords

du damier. Autrementdit, il y a toujours un point de passage à chaque bout du mur pour permettre de le contourner. Les personnages sont stockés dans une liste de personnages. Chaque personnage a une direction initialisée au hasard. Il n'y a pas de superposition sur une case : une case occupée est ou bien une pierre, ou bien la case trésor ou encore une case libre, occupée ou pas par un personnage.

## Choix de programmation

pour les choix de programmation en a tous d'abord suivi le guide pour le codage sur le sujet du projet bien sûr que en a peu à peu amélioré au départ en été vraiment sur une autre version plus compliquer à implémenter et gérer en a tous repris à zéro et refait les classes une par une au propre avec tous les getters et setters et surtout la fonction process

## Class

Cell FreeCell StoneCell SideCell TreasureCell Hunter Questionable Wall Position  
TeleportationCell MirrorCell HoleCell Graphique Control

## Etats

le programme est fonctionnelle

## Difficulté rencontrée

en a eu du mal pour le contournement des murs.

Et le déplacement des joueurs

car au départ en utiliser que une ArrayList, et après ArrayList de ArrayList qui nous a permis de mieux implémenter les choses.

## Optimisation apportée

### 1-les points bonus du cahier des charges.

En a ajouté trois classes supplémentaires qui représentent les nouvelles cases : téléportation qui renvoie le joueur dans une case libre random, la case miroir qui renvoie le joueur dans la même direction qu'il est venu symétriquement, et la case trou qui fait passer un tour à un joueur et qui reprend après son déplacement.

### 2-optimiser le programme avec plus de fonction.

En a essayé de faire le plus de fonction pour bien répartir le travail fait par chaque une et que sa soit clair

### 3-l'ajoute de l'auto initialisation des différentes cellules.

En a mis le symbole en auto initialisation sa fait économiser des lignes et vu que tel ou tel case changer pas de symbole c'est plus pratique

sur les add dans les ArrayList en a utilisé des boucles for pour minimiser les lignes de code

## Discription du jeu obtenu:

trois joueur placer a chaque fois dans des point de départ differant ce déplace chaqun leur tour pour rejoindre la case treasure en contournant les mur des que l'un des joueur arrive cest la fin de la partis, avec en parallele l'affichage

## Exemple du rendu de la parti console

dans la parti console une fois lance les joueur se deplacer automatiquement jusqu'a l'arriver d'un des troi sur la parti treasure.

Console x

maingame [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (22 mai 2020 à 08:32:16)

```
+++++
+...C...@..+
+.....+
+...#.....+
+.T.#.....+
+...#....B...+
+...#.....+
+...#...###...+
+...#....A...+
+...#.....X.+
+.....+
+.....~.....+
+.....+
+++++
```

```
personnage A :
Hunter [8 9] dir3
Case cible : 7 9
2 hurt
-> Hunter [8 9] dir 2
```

```
personnage B :
Hunter [5 9] dir7
Case cible : 6 9
best dir :4
-> Hunter [8 9] dir 4
```

```
personnage C :
Hunter [1 5] dir4
Case cible : 0 4
-> Hunter [1 5] dir 8
```

## Exemple du rendu de la parti graphique

sur la parti graphique a chaque fois qu'on appuis sur le bouton tour suivant les hunter ce deplacer d'une seul case et il faut apuiyer a chaque fois jusqu'a l'arriver de l'un des troi sur trasor

Treasure Hunter

Tour suivant

	1	2	3	4	5	6	7	8	9	10	11	12	
1	.	.	.	.	.	.	.	.	.	@	.	.	1
2	.	.	.	.	.	.	C	.	.	.	.	.	2
3	.	.	.		.	.	.	.	.	.	.	.	3
4	A		.		.	.	.	.	B	.	.	.	4
5	.	.	.		.	.	.	.	.	.	.	.	5
6	.	.	.		.	.	.	.	.	.	.	.	6
7	.	.	.		.	.				.	.	.	7
8	.	.	.		.	.	.	.	.	.	.	.	8
9	.	.	.		.	.	.	.	.	.	x	.	9
10	.	.	.	.	.	.	.	.	.	.	.	.	10
11	.	.	.	.	.		.	.	.	.	.	.	11
12	.	.	.	.	.	.	.	.	.	.	.	.	12
	1	2	3	4	5	6	7	8	9	10	11	12	

Personnage A [4 1] Dir 3  
Personnage B [4 9] Dir 1  
Personnage C [2 7] Dir 1

travail de groupe, en a eu quelque difficulté au départ mais en a vite trouver les outils et une façon de s'organiser, le travail a été répartis vraiment équitablement et en s'aider sur les partis de chaqu'un en cas de difficulté, même si en est un peu déçu du rendu car en aurai pue mieux fair en a perdu beaucoup de temp car en a pas choisie la bonne implementation mais des que en a simlifier et detailes en a pu bien avancé .