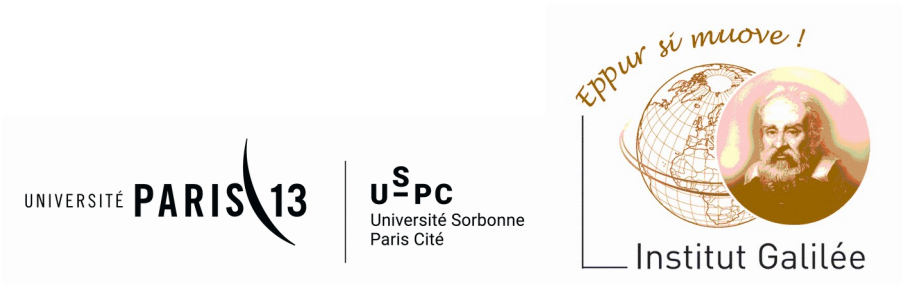

Projet Shell GRDL

Mustapha BENHALIMA 11709087
mustapha.benhalima@edu.univ-paris13.fr

Ilyes BEN YAHIA 11707486
ilyes.benyahia@edu.univ-paris13.fr

2017/2018

Algorithme utilisé :

-Chercher Thésée , le localiser et sauvegarder sa position dans le tableau.

-Chercher la possibilité de se déplacer entre les 1,2,3 et 4 autour de Thésée :

1

4 T 2

3

-Appeler la fonction chercher_S qui prend en argument la position qu'on va chercher autour d'elle et le coté (1, 2, 3 ou 4) et qui fait :

-Vérifier autour de la case, si il y a un vide dans case voisine, on appelle la fonction d'une manière récursive pour la case voisine, sinon on passe la case voisine suivante.

-Ne pas revenir sur la case qu'on a déjà vérifier en utilisant l'instruction de condition « case » .

-Si l'une des case voisine ou l'un des coté est une sortie, on sors de la fonction en affichant un message « Thésée peut réussir à sortir » .Sinon on continue à chercher jusqu'au où Thésée est bloqué entre des murs et des minotaures , si il trouve pas , on afficher « Thésée ne peut pas réussir à sortir ».

-La fonction cherche_T cherche d'abord la position de Thésée, puis elle fait l'appel de la fonction cherche_S pour commencer à chercher la sortie à partir de la position de la position de Thésée.

-La fonction MT_labyrinthe initialise les minotaures et Thésée d'une manière aléatoire dans le labyrinthe dans les case vide (de valeur 1).

-La matrice contient les valeurs suivantes :

« 0 » : mur

« 1 » : vide

« 2 » : minotaure

« 3 » : Thésée

« 4 » : entrée/sortie

-Pour l'affichage :

« MM » : minotaure

« TT » : Thésée

« [] » : mur

« ES » : entrée/sortie

« » : vide

-