P1_DUH_01_1
Leon Brandt
Samy Bettaieb
Hadrien Allegaert
Alexandre Couchard

Rapport de mission

Introduction :

Ce projet est une suite de l'APP0. Il était demandé de vérifier des codes déjà écrit et de créer des terrains qui les mettaient en échec. Nous avons choisis de travailler sur le code de la recrue A. Nous verrons ci-dessous la carte qui met en échec son code et sa correction.

Description:

La faute repérée pour faire planter l'algorithme se trouve dans les lignes 10-11 et 24-25. Elle ne permet pas de vérifier la présence d'un loup avant un mouvement ce qui pose problème à notre espion qui n'aime pas trop être dévoré vivant.

Pour comprendre la faute, il faut décomposer l'algorithme.

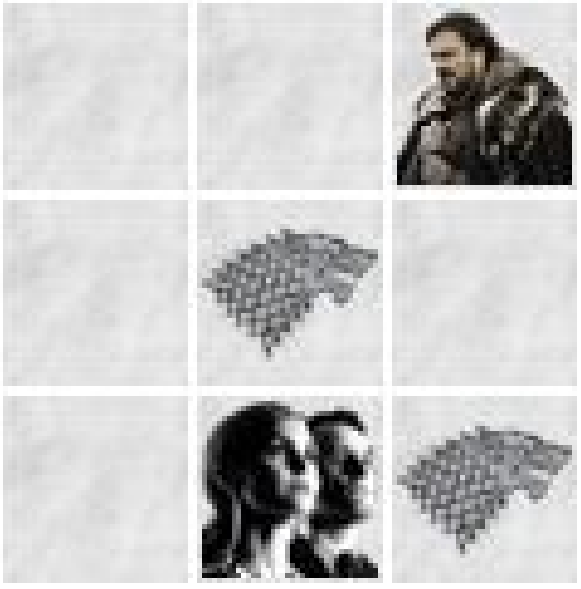
Dans les lignes 1-7, une boucle s'ouvre et se termine quand on est sur la bonne colonne (celle ayant la même valeur d'abscisse ("X") que Ned).
Après l'algorithme détermine de quel côté de Ned les espions se trouvent (quelle valeur de X par rapport à celle de Ned) et on tourne la figure dans le bon sens pour partir dans cette direction.
Après s'être tourné dans la bonne direction, une condition détermine s'il y a un loup devant et si le chemin est libre, et si c'est le cas on bouge. Si cette condition n'est pas remplie, on se retrouve à la ligne avec le problème. Le personnage se tourne vers la gauche et une autre condition prend place. L'auteur de l'algorithme a mis « can_move » mais il a oublié de mettre « is_in_front_of_wolf ». Si cette condition est remplie, le personnage bouge.
Donc si, il ne se trouve pas encore sur la colonne de Ned et qu'il y a un obstacle quelconque devant lui (Arbre/Loup), il se tourne vers la gauche. Là, il teste s'il y a un arbre ou un bord, si ce n'est pas le cas, il avance d'une case. S'il se trouve avant un loup, il ne le voit pas avec la condition, il avance et finit par se faire manger par un loup

P1_DUH_01_1
Leon Brandt
Samy Bettaieb
Hadrien Allegaert
Alexandre Couchard

Algorithme A corrigé

```
1       while get_x() != get_target_x():
2               if get_x() < get_target_x():
3                       while get_direction() != EAST:
4                               turn_left()
5               else:
6                       while get_direction() != WEST:
7                               turn_left()
8               while is_in_front_of_wolf() or not can_move():
9                       turn_left()
10                      if can_move() and not is_in_front_of_wolf():
11                              move()
12                               turn_right()
13              move()
14
15      while get_y() != get_target_y():
16              if get_y() < get_target_y():
17                      while get_direction() != SOUTH:
18                              turn_left()
19              else:
20                      while get_direction() != NORTH:
21                              turn_left()
22              while is_in_front_of_wolf() or not can_move():
23                      turn_right()
24              if can_move() and not is_in_front_of_wolf():
25                      move()
26                      turn_left()
27              move()
28
29      spy_on_target()
```

P1_DUH_01_1
Leon Brandt
Samy Bettaieb
Hadrien Allegaert
Alexandre Couchard



Conclusion :

Pour conclure, nous avons trouvé une faille dans l'algorithme de la recrue A, et utilisé cette faille pour mettre en place un terrain qui fait dévorer les espions par un loup.
L'algorithme présentait le défaut de ne pas vérifier avant chaque "move()" si oui ou non il n'y avait pas de loup en face de lui, poussant donc les espions à se faire dévorer dès qu'ils devaient éviter un obstacle en le contournant.

Puis nous avons corrigé l'algorithme de manière à passer le terrain choisi. Nous avons simplement choisis de rajouter une condition dans les boucles qui présentaient le défaut de ne pas vérifier le booléen "is_in_front_of_wolf == FALSE" avant son "move()".