AP3 - PROJET BATTLEROBOT

01 - Création de l'intelligence artificiel

- Explication de la solution : marquage par craie

Le Hero perdu dans le labyrinthe ne connaît que des Salles autour de lui et auxquelles, il a accès. Ainsi pour trouver la Salle de sortie, il n'a d'autre choix que de parcourir l'ensemble du labyrinthe jusqu'à trouver son objectif. Il doit reconnaître les chemins inconnus des chemins où il est déjà passé, et des chemins qui mène à un cul de sac. On appellera cela un « coup de craie » fait par le Hero : lors des ces choix, il sait où il peut aller où non. En se référant au nombre de Salle accessibles, on distingue deux cas possibles :

- 1 salles accessibles : signifie que Bob se trouve dans un cul de sac, donc dans salle qui doit être interdite d'accès. Ainsi il remonte jusqu'au prochaine croisement pour indiquer ce chemin comme interdit.
- 2 à 4 salles accessibles : il marque toute les salles visitées comme « passées », il regarde les salles autour et leur marquage pour savoir laquelle emprunter. Si une salle est « inconnue », c'est a dire qu'elle n'a aucun marquage, il y entre et la marque comme « passée ». Sinon si toute les salles sont marquées, il va uniquement emprunter celles qui ne sont pas « interdites ».

- Illustrations et schémas

On pourrait simplement imaginer le marquage du Hero de cette manière : il peut compléter ses marquages en ajoutant un trait (voir Dessin1).

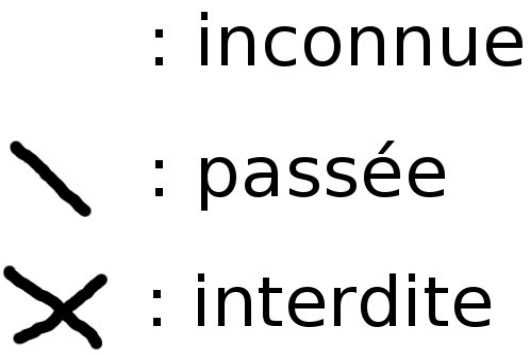

Illustration 1: Marquages de craie

Pour mieux voir ça, voici un exemple où le Hero parcours le labyrinthe et se retrouve à un croisement (voir Dessin 2) :

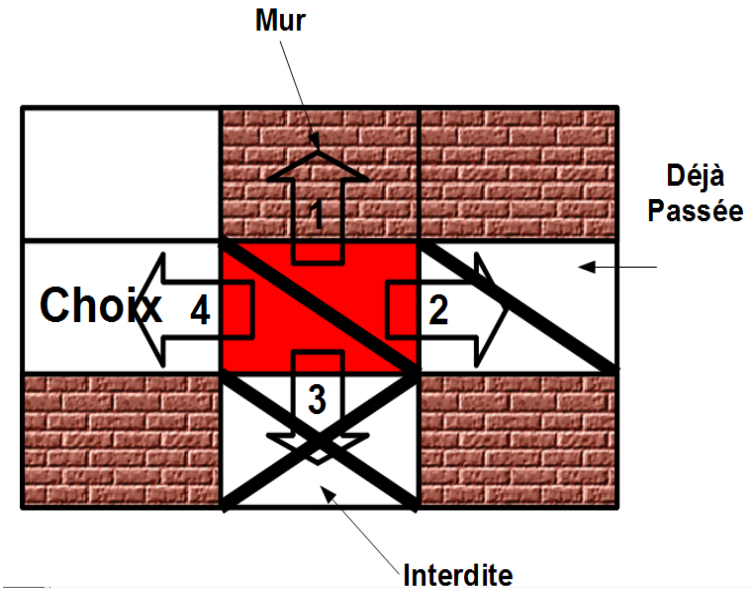

Illustration 2: Exemple d'un croisement

- le choix 2 : il est déjà passé par là, il va voir si il y a pas une salle inconnue
- le choix 3 : est marqué comme interdit, donc il ne la prend pas en compte
- le choix 4 : est inconnu, il va donc emprunter ce chemin sans voir les autres salles possibles

Ici on prend l'exemple d'un petit labyrinthe, où on a essayé de représenter les marquages qu'il effectue et les décisions qu'il prend durant son parcours (voir 3 2). la salle jaune est l'entrée et la salle verte est la sortie.

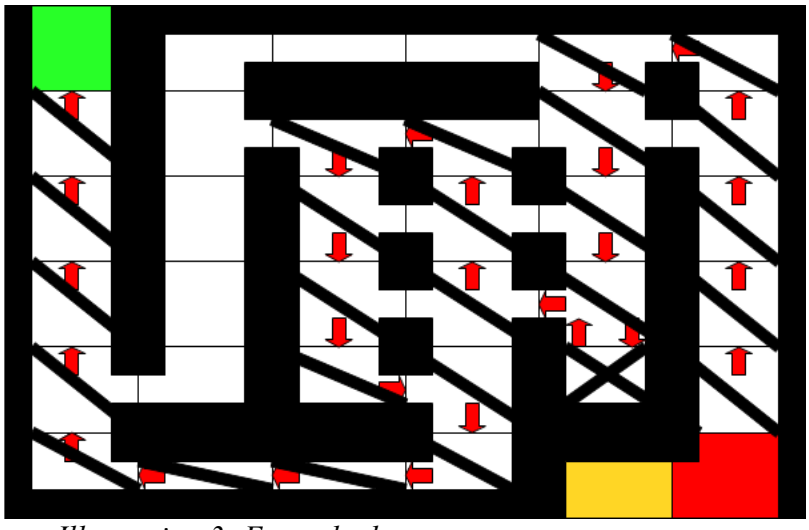

Illustration 3: Exemple de parcours

Par contre si il fait le parcours dans le sens inverse, il ne prendra pas le même chemin (voir Dessin 4 3)

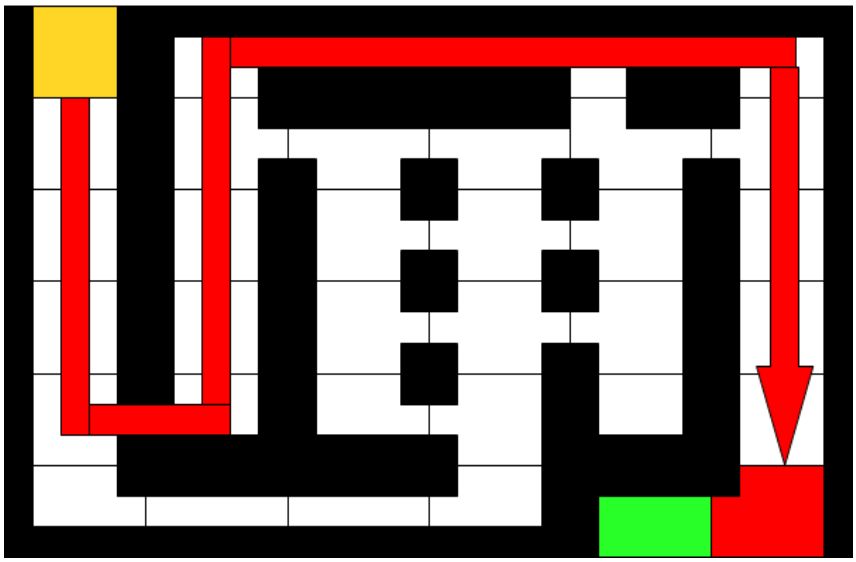

Illustration 4: Exemple du parcours inverse

- Détail et algorithmique de la classe

Quelques variables et méthodes ont été rajoutées à la classe « MyPersonnage.java », pour mieux répondre à notre idée dite des « coups de craie ».

Rôles des variables :

Les Collections de Salles « passed » et « forbid » représentent la trace des marques mise dans chacune des salles visitées par le Hero. Quand il rentre dans une salle inconnue, on ajoute la salle dans la collection passed. Lorsque la Collection « sallesAccessibles » contient qu'une seule Salle, cette dernière est ajoutée dans « forbid ».

Une autre Collection de Salles « allowed » permet de faire une copie de « sallesAccessibles » et d'y appliquer une mise à jours pour y retirer toutes les salles qui sont présentes dans « forbid ». Cela évite de faire une vérification à chaque fois qu'on fait le choix dans la prochaine Salle à retourner.

Méthodes de classe :

La méthode « salleAllowed » prend en paramètre la collection contenant toute les salles accessibles et renvoie une Collection de ces salles accessibles en enlevant les salles interdites. On l'appel uniquement au début de la méthode « faitSonChoix » pour enregistrer la nouvelle collection « sallesAccessibles » mise à jour dans « allowed », comme expliqué précédemment.

La méthode « isPassed » prend en paramètre une Salle et retourne un booléen pour dire si cette salle appartient à la collection « passed » ou non. On fait appel à elle pour chercher les salles non présentent dans « passed » afin d'indiquer la prochaine salle à emprunter.