Rapport d'expérimentations

AVAST

Choix d'algorithme

Au départ, l'IA implémenté ne faisait aucune action. Il vérifiait juste s'il pouvait guérir une maladie (StupidAi). Avec cette IA nous avions 1% de victoire.

Nous avions ensuite ajouté les déplacements vers une zone où il y avait le plus de maladies. L'IA cherchait le plus court chemin pour atteindre un foyer d'infection (PcchAi). La méthode la plus instinctive a été de regarder les différents chemins possibles pour atteindre un foyer d'infection et trouver celui qui a le plus faible coût. Mais cette méthode s'est avérée impossible car il y avait beaucoup trop de combinaisons possibles donc l'IA tournait indéfiniment.

Par la suite, nous avions implémenté une IA qui calculait la distance euclidienne entre la ville actuelle et les villes ayant un foyer d'infection à l'aide des coordonnées du graphe XML (CheatAi). Et l'IA se déplaçait alors sur la ville la plus proche de la ville ayant un foyer d'infection.

Au final on a utilisé l'algorithme « Dijkstra » vu en cours de Graphes en L3 pour trouver plus rapidement le plus court chemin entre la ville actuelle et les foyers d'infection (DijkstraAi).

Choix de patron pour le Gui

On a retenu le choix du patron de comportement « Observer » vu en cours de Systèmes d'Information Avancés. Il y a un objet qui observe les modifications d'un autre objet. L'objet observateur est le GUI et l'objet observé est l'état du jeu (Board). Dès qu'il y a une modification de l'état du jeu, par exemple le déplacement du joueur, l'observé va notifier ses observateurs du changement via la méthode « notifyObserver ». Ainsi l'observateur va chercher les informations de l'observé et mettre à jour l'affichage du GUI.