Réflexion algorithmique pour l'IA

Réflexions et stratégie

Après l'implémentation des fonctions et de notre structure de donnée nous avons réfléchis à une stratégie qui est à la fois défensive et offensive.
Nous appellerons cette stratégie : La Meute et loup solitaire.
Elle consiste à encercler notre alpha par les autres loups et envoyer l'Omega en loup solitaire.

La meute

L'aspect défensif :

La meute se déplace en groupe avec l'alpha au milieu pour qu'il ne soit pas facilement attaquable par l'adversaire.
Vue qu'un loup devenant humain occupe toujours la case et agit donc comme une barrière contre l'ennemi.
A chaque tour de jeux la meute avance vers l'alpha ennemi.

L'aspect offensif :

La partie offensive réside dans le fait que les loups avançant en « meute » pour attaquer, ils bénéficient donc de beaucoup de bonus des loups allié autour.
De plus si un loup dois se nourrir alors meutes pivote afin de placer le loup qui doit se nourrir dans la direction de la nourriture afin de réduire le temps et le déplacement de ce dernier.

L'Omega

L'Omega lui seras entrain de parcourir le plateau de jeu à distance de la meute en attaquant et en pacifiant au maximum. Le but étant de le faire se nourrir au maximum afin qu'il épuise les ressources de nourritures proches des loups ennemis.

Algorithmique

Nos algorithmes contiendront beaucoup de vérifications afin de tester la présence de certains éléments dans un certain rayon.

Phase de déploiement / Phase de mise en boule

La meute

1) Ecarter l'alpha d'au moins 3 cases des bords de table
2) Placer les loups autours

L'Omega

1) « Fuite » de l'omega vers la plus grosse concentration de loups adverse en restant le plus loin possible de la meute.

Phase de chasse

La meute

1) Localisation et traque de l'alpha adverse pour l'attaquer.

L'Omega

1) L'omega reste a plus de 6 cases de la meute.
2) Pacifie dès que 3 adversaires à portée
3) Va se nourrir après 2 pacification

Proposition des fonctions générale d'actions, seront très certainement complètées par des petites fonctions utilitaire de pathfinding :

def far_from_walls()
=> get a position from 3 from"walls" for the alpha
def pack_empty_space()
=> get the nearest empty position around the alpha
def away from_pack()
=> Move omega far as possible (7 spaces) from pack
def target_alpha()
=> Target and get alpha ennemy's position
def feed_the_pack()
=> When energy of the pack below 50%
def pacifeed()
=> pacify when appropriate
=> Feeds when need