Algorithme Génétique pour le Core War

Principe:
Permettre la création de fichiers RedCode, générés de manière génétique. L'algorithme génétique va permettre de partir d'un fichier vide puis suivant le nombre de générations que l'on souhaite avoir, obtenir un code avec une certaine stratégie et une cohérence entre les différentes instructions mises dans le fichier.

Théorie:
On cherche à créer une forme qui peut se déplacer sur un axe x. Cette forme doit pour gagner atteindre un point qui se trouve sur l'axe x, à une certaine distance, avec des obstacles de toutes formes sur son chemin. Le but de notre algorithme est donc de pouvoir s'adapter au différents événements qui pourraient arriver. Pour se faire, nous partons de zéro. Dans un premier temps, nous allons créer des listes et des variables capables de contenir toutes les informations possibles sur notre forme. Une liste de formes (cercle, rectangle, carré …), une liste d'actions (sauter, aller vers la droite, aller vers la gauche …), une variable de taille, et une variable de vitesse.

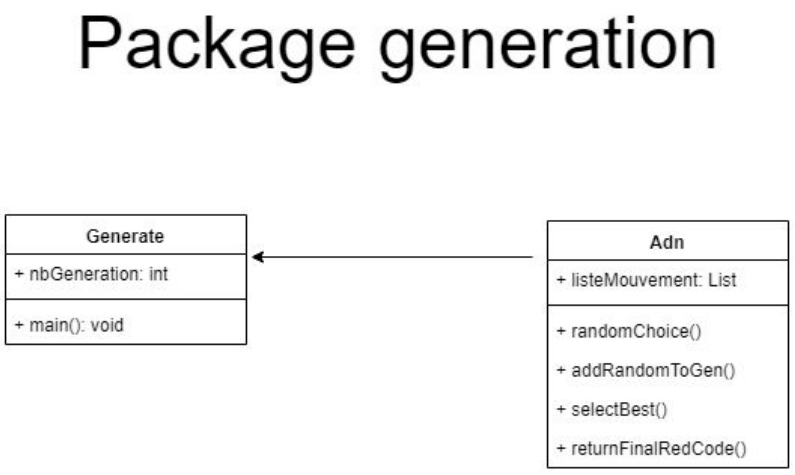
Pour la première génération, nous allons créer aléatoirement un groupe de formes (une dizaine ou une centaine), chacune des variables de chaque forme sera générée aléatoirement (une forme prise parmi la liste, une taille et une vitesse choisie aléatoirement). Ensuite, on va créer une fonction d'évaluation qui va donner un score à chaque forme qui sera apparue. Une fois la génération terminée, nous mettons un temps impartis et à la fin de ce temps-là. Nous mettons un score à chaque forme en fonction de son état d'avancement vers la destination. Une fois le temps écoulé, on prend un pourcentage des meilleurs formes qui sont allées le plus loin par exemple 30 %. Ensuite, on prend l'ADN de ces formes (la description des variables), et on injecte cet ADN dans une nouvelle génération. Mais on ne mettre pas toute la génération avec l'ADN des gagnants, nous le mettrons sur 50 % de la nouvelle génération, le reste de la génération sera composée d'une partie de l'ADN générée aléatoirement et une partie de l'ADN gagnante avec des mutations sur certaines variables, et les but est de répéter cette opération n fois. À la fin, si l'algorithme est bien programmé alors on devrait avoir une forme qui s'est adaptée au terrain et qui permet d'aller du point A au point B le plus rapidement possible.

Pratique:
Maintenant le but est d'adopter cette technique à notre Core War. Dans un premier temps, nous allons devoir créer pour la première génération plusieurs fichiers RedCode (par exemple 50). Ensuite, on va leur attribuer à chacun des valeurs tirées au hasard (les opérateurs MOV, DAT …) puis aussi tirer au hasard les adresses de départ et d'arrivée (qui sera au maximum de la longueur de la mémoire), ainsi que les signes ($, #). Ensuite, il faudra les faire combattre dans notre Core War contre un programme que l'on aura défini.

Nous prendrons les gagnants des différentes batailles et nous nous en servirons pour générer les générations suivantes sans oublier de mettre une partie de hasard à chaque génération.

Diagramme UML:

Voici le diagramme UML que j'ai imaginé, il sera amené à évoluer



But recherché:

Avoir un algorithme de création génétique qui permet la création de fichiers RedCode avec une stratégie de jeu, et non pas seulement un fichier RedCode avec seulement des instructions les unes après les autres sans liens.