PROJET D'ALGORITHMIQUE

Le Wally est un robot nettoyeur produit par la société Xarpi. Eveuh, une jeune peintre pleine de talent, est intéressée par l'achat d'un Wally pour son salon. Cette dernière, adepte d'un style de peinture assez original, le *foot painting*, laisse régulièrement de longues traînées de peintures derrière elle et n'en peux plus de voir le sol de son atelier se colorer après chaque tableau.

La mission du Wally sera donc de nettoyer toute trace laissée par Eveuh sur son carrelage. Sa vision n'étant pas très performante (il ne peut voir qu'à une courte distance devant lui), la société Xarpi va utiliser un algorithme génétique pour rechercher la meilleure stratégie à adopter : il faut nettoyer un maximum de traces tout en optimisant les déplacements du robot pour économiser sa batterie. Une des propriétés intéressantes de ces traces de peintures est qu'elles forment une piste généralement continue sauf en quelques points où Eveuh marche à cloche-pied (mais jamais plus de 3 carreaux d'affilée).

La société Xarpi nous commande le développement d'un logiciel permettant de faire évoluer l'intelligence artificielle du Wally en utilisant un algorithme génétique. Ce logiciel devra aussi permettre de tester et de visualiser le comportement d'un Wally dans un salon virtuel (une partie du code nécessaire est déjà fournie, vous pouvez le modifier à votre gré).

Petit détail d'importance, le salon d'Eveuh est équipé, comme tous les salons modernes, de murs téléporteurs. Ça agrandit l'espace, et c'est pratique pour faire du footing. En d'autres termes, le plateau est un tore.

Bonus : le salon peut contenir des meubles inamovibles.

COEUR DE L'APPLICATION

Le salon d'Eveuh sera représenté par un plateau carré, chaque case pouvant contenir de la peinture, un obstacle ou rien. Des structure Board et Robot sont déjà fournies, vous pouvez néanmoins les enrichir.

Représentation de l'IA des Wally

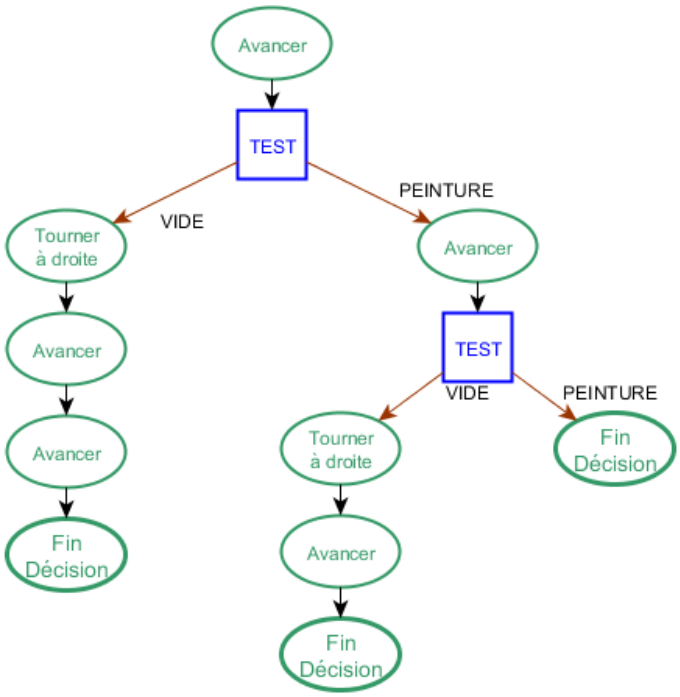
Un algorithme génétique manipule des populations d'individus décrits par leur ADN. Il nous faut donc trouver un moyen de représenter le comportement des Wally sous la forme de séquences "ADN" que l'algorithme pourra évaluer, sélectionner, faire muter et croiser.

Un système de prise de décision simple peut être représenté par un arbre. En effet, le comportement suivant :

```
avancer, Que vois-je ?
      Si Vide alors : tourner droite, avancer, avancer, fin décision;
      Si Peinture alors : avancer, Que vois-je ?
                  Si Vide alors : tourner droite, avancer, fin décision;
                  Si Peinture alors : fin décision;
```

N.B. : l'instruction "fin décision" signifie en réalité "recommencer la routine de prise de décision depuis le début"

peut être représenté ainsi :



La **population initiale** sera constituée de Wally ayant chacun un ADN (des arbres **générés aléatoirement**). Vous veillerez cependant à borner le nombre maximal de nœuds.

Croiser deux comportements revient alors à **sectionner** les arbres correspondants en **un point aléatoire** chacun, et échanger les branches (voir schéma en annexe).

Une **mutation** survient lorsqu'un noeud se change aléatoirement en un autre (voir schéma en annexe).
Si le noeud obtenu après la mutation a un nombre de branches supérieur à l'original, alors les descendants initiaux sont affectés à une branche quelconque du nouveau noeud, et les autres branches sont générées aléatoirement (comme lors de la création des premiers individus).
Si au contraire le nombre de branches diminue, alors des branches aléatoires sont "effacées".
Une mutation peut également insérer un nœud n'importe où, même à la fin d'une branche, ou supprimer un nœud existant.

L'évaluation d'un individu se fera à travers une série de simulation. Une interface graphique est fournie afin de visualiser un individu particulier. Cette visualisation ne doit néanmoins pas être utilisée pour tous les individus générés (car cela est trop coûteux).
Les grilles `trail_0` à `trail_5` (fournies) seront successivement utilisées pour estimer l'efficacité du sujet testé à nettoyer le maximum de peinture en un temps donné, tout en évitant autant que possible de se déplacer pour rien. Elles sont accessible via la fonction

```
loadTrail(V numTrail as Integer) as Board
```

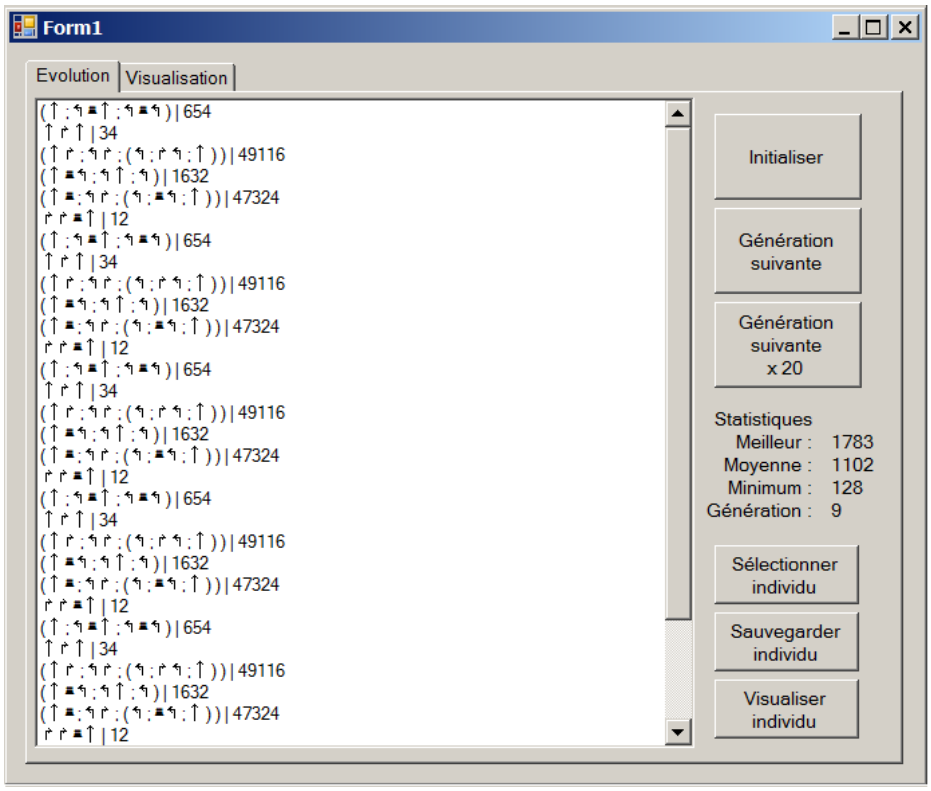
Toutes les actions (avancer, tourner et observer) prennent une seconde en temps simulé, et le robot dispose d'un temps limité pour montrer ce qu'il sait faire.

INTERFACE

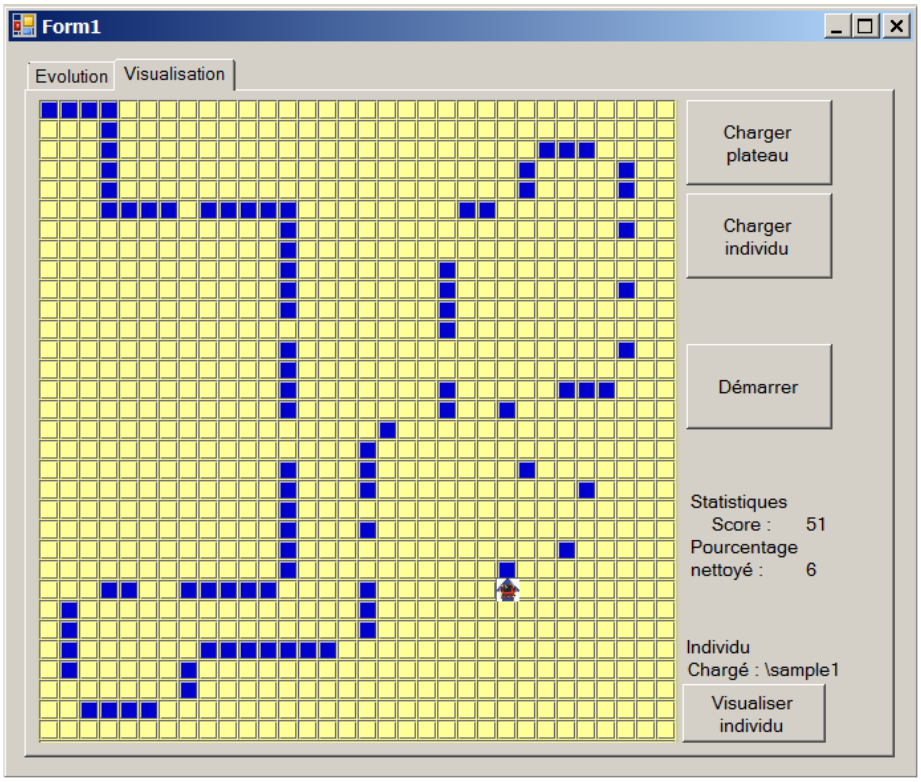
L'interface sera composée de deux panneaux (accessibles par des onglets) : le panneau évolution et le panneau simulation.

Onglet évolution

Ce panneau offre l'accès aux commandes principales de l'algorithme génétique. Il permet de générer une population initiale de Wally, de la visualiser et de faire avancer l'algorithme génération après génération. Il devra aussi présenter des statistiques sur l'évolution en cours et permettre la sauvegarde d'un individu se



Onglet simulation



Livrables

Le code source de votre programme sera accompagné d'un document de 5 pages maximum, où vous décrirez brièvement les choix que vous avez faits, ainsi que le comportements des principaux sous-programmes.

Un fichier .zip contenant ce document et le code source du projet devra être déposé sur la plate-forme pédagogique au plus tard le Lundi 12 Mai.

Ce qui est demandé

Ce projet est prévu pour être évolutif. Il est ainsi découpable en plusieurs niveaux de difficultés. Un barême est fourni à titre indicatif.

Niveau 1 (10 pts)

- Vous arrivez à générer un individu, c'est à dire créer son ADN (un arbre aléatoire) et à l'évaluer, c'est à dire lui donner une note (fitness) qui sera proportionelle à son efficacité (a nettoyer la peinture en un nombre d'actions aussi petit que possible).
- Vous êtes capables d'exécuter l'ADN d'un robot et de visualiser le comportement ainsi créé (en action) dans la fenêtre de visualisation, à l'aide de la fonction de dessin fournie.

Niveau 2 (3 pts)

- Vous pouvez générer une population initiale d'individus aléatoires, les évaluer, afficher leurs ADN et leurs fitness dans une listBox ainsi qu'afficher des statistiques (fitness max, min, moyenne) sur la population courante.
- Vous pouvez sélectionner un individu parmi la population courante et lancer la visualisation de sa simulation dans l'onglet "visualisation".

Niveau 3 (7 pts)

- Vous êtes capables de croiser les ADN de deux individus et d'appliquer des mutations (changer un noeud, insérer un noeud, supprimer un noeud) sur cet ADN
- Votre algorithme génétique est fonctionnel et actionnable via l'interface de l'onglet "Evolution" : on peut générer une nouvelle population et itérer l'algorithme génétique sur une ou plusieurs générations successives. La fitness max de la population s'améliore au fil des générations.
- Vous pouvez charger et sauvegarder un robot.

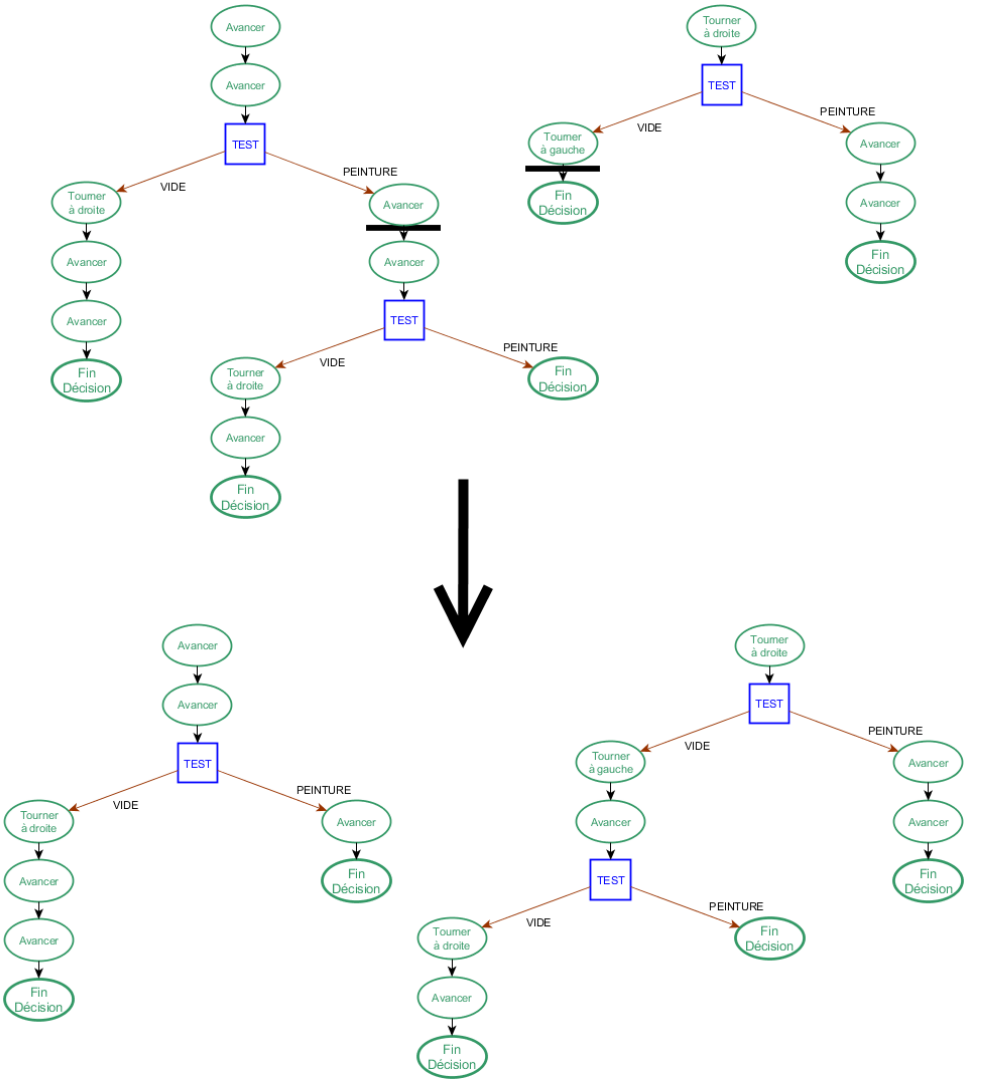
Niveau 4 ("On verra" pts)

- Résultats : vos robots sont de plus en plus efficaces au fil des générations. C'est-à-dire que vous avez trouvé une fonction d'évaluation prenant en compte à la fois l'efficacité du robot à nettoyer les traces de peintures et sa capacité à nettoyer la pièce en un minimum de temps. Les robots au "sommet de l'évolution" (meilleurs robots, obtenus après plusieurs générations) se montrent relativement intelligents.

N'oubliez pas de sauvegarder vos meilleurs échantillons pour ne pas avoir à refaire une évolution le jour de la soutenance.

Annexes

Croisement



Mutation