19/06 :

Premier jour, présentation du projet par les encadrants, plusieurs choix s’offrent à nous. Nous pouvons soit utiliser une interface qui a été faite par un groupe l’an passé sur netlogo et donc nous focaliser uniquement sur les paramètres de modélisation et leurs impacts. Il est également possible de recréer notre interface graphique sur netlogo ou processing. Jugeant que la tâche de créer l’interface graphique et la simulation n’était non seulement pas insurmontable mais qu’elle était surtout intéressante à faire pour apprendre à programmer. C’est pourquoi nous avons pris la décision de tout programmer nous-même sur python en utilisant la bibliothèque pygame pour l’interface graphique utilisateur.

Une fois le choix de l’approche, nous avons établi un cahier des charges pour savoir la direction vers laquelle nous allons mais surtout pour nous assurer d’avoir bien compris le sujet ainsi que ses enjeux, et donc ce que nous cherchions à démontrer/comprendre.

Voici donc les éléments qui nous semblaient les plus importants :

-des spots de nourritures dont nous décidons la position, l’attractivité, le nombre, la qualité de la nourriture

-des agents, qui possèdent une énergie, un type (altruiste, tricheur, basique), des gènes, une capacité à se reproduire (seuls ou à plusieurs), une “intelligence” (aller vers les phéromones/la nourriture)

En parallèle nous avons établi l’ensemble des éléments que l’utilisateur (nous) peut modifier :

-part d’altruistes/basiques/tricheurs au début de la simulation

-nombre de spots de nourritures, quantité de nourriture par spots, attractivité de la nourriture

-énergie de base des agents, énergie nécessaire à la reproduction, coût en énergie de la reproduction

-aging process, combien d’énergie les agents perdent en fonction de leur âge à chaque itération

-puissance des phéromones

-probabilité de mutation des gènes

-choix du gène dominant entre altruiste/tricheur (le gène basique est totalement récessif pour qu’il ne puisse pas proliférer)

Nous avons également pensés à certains ajouts qui pourraient être faits par la suite :

-collisions entre agents

-topologie de la nourriture (empoisonné → les altruistes préviennent, les autres non)

Une fois les tâches déterminées, nous avons séparé le travail comme suit :

-Achraf et Chems s’occupent du déplacement des agents

-Romain s’occupe de l’interface graphique et de coordonner l’univers (implémentation entre les différentes classes)

-Joseph s’occupe des spots de nourriture

-Yanis s’occupe de la reproduction des agents (seul ou à plusieurs) et de l’alimentation des agents

Dès lors, la programmation commença.

20/06 :

Ayant bien segmenté le travail, nous pûmes commencer à intégrer les différentes classes dès le deuxième jour. La première simulation de déplacement des agents suivaient une trajectoire rectiligne uniforme et dès que l’agent arrive face à un mur alors il change aléatoirement de direction. Très vite Achraf s’est penché sur la question d’un mouvement plus “naturel” des agents, il a essayé de faire une marche “aléatoire”, ce qui s’est révélé compliqué. Premièrement, toutes les classes se sont retrouvées dans un même fichier, après réflexion, nous avons fait un fichier par classe pour mieux pouvoir modifier et comprendre le programme. Étant donné que nous avions beaucoup appuyé la veille sur l’importance d’un code commenté (en anglais), il n’était pas compliqué de comprendre ce que les autres avaient fait. Après le passage et les conseils de monsieur Dessalles, nous avons mis toutes les constantes du programme dans un fichier json pour que l’on puisse modifier les constantes simplement avant de lancer le programme. Pour éviter les répétitions dans les différentes classes, une classe mère “entity” a été créée.

Nous avons passé l’après-midi à mettre en commun ce que nous avions fait la veille au soir et le matin pour avoir une ébauche de simulation fonctionnelle. Le travail de Chems et Romain le soir ont permis d’avoir cette embauche fonctionnelle.

21/06 :

Le matin, nous avions donc des agents capables de se déplacer et globalement de se reproduire. Il y avait beaucoup de bugs mineurs et donc la plus grande partie de la journée a été utilisée pour les résoudre. De plus pour l’instant nous n’avions que les agents basiques, l’implémentation des autres agents fût compliquée dans le cas des altruistes à cause des phéromones. Nous avons donc repensé l’architecture de nos classes et de notre programme pour pouvoir simplifier la compréhension, le fonctionnement et le débogage. En milieu d’après-midi nous avions donc une simulation totalement fonctionnelle et nous avons commencé à modifier les paramètres pour obtenir différents scénarios. Pour être sûr que la simulation fonctionne nous avons regardé si des agents basiques pouvaient proliférer, et quel que soit le nombre d’agents basiques à l’état initial, leur population finie par disparaître. Ceci étant bon nous avons joué sur les paramètres de la nourriture pour que les agents altruistes soient nécessaires à la pérennisation de l’espèce. Voici un scénario type et les observations :



On observe bien que la population des agents basiques finie par disparaître. La proba de mutation étant de 20%, on a apparition des agents profiteurs qui prolifèrent très vite, cette observation est logique car les altruistes perdent de l’énergie lorsqu’ils libèrent des phéromones donc il leur reste moins d’énergie pour se reproduire, tandis que les profiteurs sont attirés par les phéromones donc profitent des altruistes sans que ça ne leur coûte rien. Cependant cette prolifération est stoppée, en effet les profiteurs consomment plus donc les altruistes ont moins donc ils se reproduisent moins donc il y a moins d’altruistes pour libérer des phéromones, donc les profiteurs ont plus de mal à se nourrir donc se reproduire. Ce point de bascule est très remarquable aux alentours de 2500 itérations mais on peut également l’observer vers 7500, 8500, 9500 et 11500 itérations.

Prolifération des profiteurs 🡪 Moins d’altruistes 🡪 Moins d’accès à la nourriture 🡪 Moins de profiteurs 🡪 Prolifération des altruistes 🡪 Prolifération des profiteurs. On a bien un phénomène cyclique. La question est maintenant de savoir si l’on peut prédire ce point de bascule en fonction des différents paramètres de la situation. Nous avons également observé que moins il y a de nourritures dans l’univers plus les altruistes sont nombreux et nécessaires à la survie de l’espèce (observation totalement logique mais qui nous laisse à penser du relatif bon fonctionnement de notre modèle).

22/06 :

Nous avions pour l’instant un seul gène, celui correspondant au type de l’agent, la probabilité de mutation étant fixe il y avait une incohérence : si un agent altruiste a un enfant, celui-ci sera un altruiste avec une proba p (p relativement élevée) et sera profiteur avec une proba 1-p (faible). L’incohérence réside dans le fait que si l’enfant mute et est donc un profiteur alors s’il se reproduit, son enfant aura de grande chance d’être lui-même profiteur, ce qui semble bizarre étant donné que ses ancêtres (hormis son parent) sont altruistes. Nous avons donc rajouté deux nouveaux gènes pour pallier à ce problème, le gène de proba de mutation, ainsi, la proba de mutation n’est plus fixe mais est propre à l’individu, nous n’avons pas encore vraiment trouver une bonne modélisation de son changement au fil de la reproduction, pour l’instant la valeur du gène de l’enfant est égale à celle du parent plus une certaine valeur prise aléatoirement sur une petite plage de valeur de manière uniforme. L’ajout le plus intéressant est celui du gène type, c’est une liste de n valeurs (pour l’instant 10 mais c’est modifiable) prenant 1 ou 0, 1 correspond à l’altruisme et 0 à la triche. A l’instant initial, le gène des altruistes est uniquement constitué de 1 et ceux des profiteurs de 0 (les basiques n’en ont pas car leur reproduction n’est pas soumise aux mutations). Lors de la reproduction, chaque valeur du gène à une probabilité de muter (passer de 1 à 0 ou inversement) égale à la valeur du gène proba du parent. Ensuite le type de l’enfant sera pris de manière uniforme sur ce nouveau gène, ainsi un parent très altruiste, c’est-à-dire avec de nombreux 1 dans son gène type (exemple 9), si après mutation son enfant a 8 1 dans son gène type alors il aura une proba 8/n (ici 8/10) d’être altruiste. Et s’il devient profiteur (proba 2/10), son gène type étant « très altruiste », s’il a un enfant, celui-ci sera plus probablement altruiste que profiteur.

26/06 :

Rajout de tigres, on sait pas si ça va travailler la simulation.

Réunion avec Dessales nous a permis de savoir qu’il faut trouver un moyen de faire vivre les altruistes, par exemple implémenter un lien de famille comme solution pour les altruistes

* Implémentation du script de tests, en choisissant un paramètre à tester le temps, la plage de valeurs à tester et la durée de la simulation (nombre d’étapes)
* Avancement dans le poster
* Début d’implémentation de l’option de déplacement en famille pour les altruistes, ce privilège intra espèce on l’appelle

*Kin selection* :

*Definition* : kin selection is a process whereby natural selection favours a trait due to its positive effects on the reproductive success of an organism's relatives