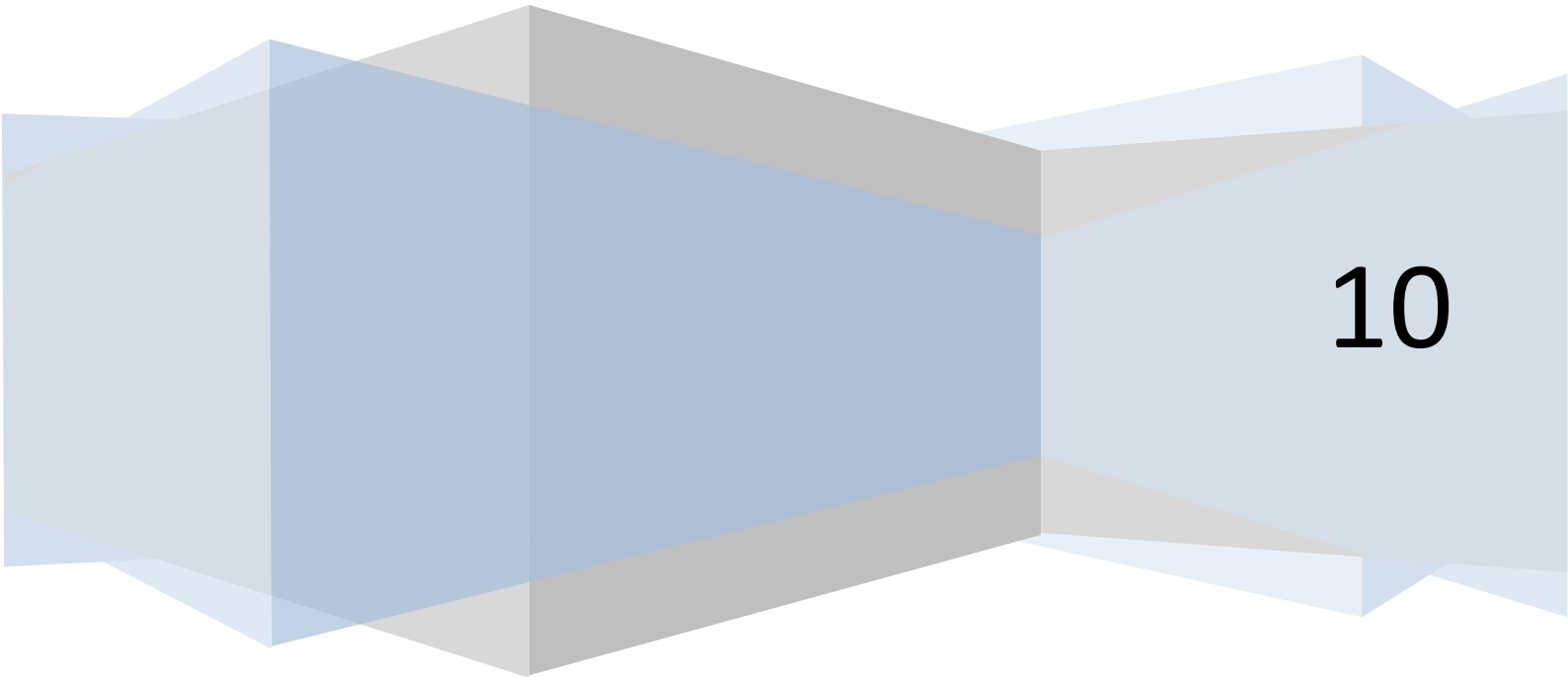


Projet : EVOLUTION

Programmation en C++ et UML

Bruno GABRIEL- Xavier MARTIN



10

Sommaire

Introduction :	3
Univers d'évolution :	4
Etre vivant :	5
Les fonctions :	5
Le déroulement	8
Conclusion :	9
Annexe :	10

Introduction :

Lors de ce projet il nous ait demandé de réaliser un programme dans le langage C++. Grâce à l'utilisation de ce nouveau langage orienté objet nous allons devoir réfléchir différemment que durant les précédents projets de programmation. En effet l'utilisation de l'objet diffère fortement de ce que nous connaissions jusqu'à maintenant. Ce projet consiste en la réalisation d'une application simulant un univers possédant de l'herbe, des moutons et des loups, avec des fonctions spécifiques pour chacun des ces objets. On peut d'hors et déjà penser que ces êtres vivant devront posséder un semblant d'intelligence pour vivre, ce qui nous amène à l'intelligence artificielle que nous essayerons d'inclure à cette simulation.

Univers d'évolution :

L'univers que nous avons choisie de réaliser en 2D est un plateau (tableau) de taille 10x10 sachant que le haut rejoint le bas et la gauche rejoint la droite. Notre plateau sera donc un tableau 2D nommé « world » et de type « monde ». « monde » étant une classe possédant comme attribut le nombre d'être vivant et de sels minéraux.

Chaque case de ce tableau peut donc contenir plusieurs être vivant à chaque moment, au maximum 3, de l'herbe puis un ou deux animaux ainsi qu'un potentiel en sels minéraux.

Lors du lancement de cette application nous avons laissé le choix à l'utilisateur du nombre de tour (un tour correspondant à 1 mois) qu'il souhaite voir évoluer l'univers. Ce choix est compris entre 2 et 50, il nous a semblé que cela était suffisant.

Il peut de même choisir le nombre de mouton (inférieur à 20) et le nombre de loup (inférieur à 20) qu'il souhaite voir évoluer. Quant aux nombres de case avec de l'herbe nous lui laissons encore une fois le choix et celui-ci est limité à 80 (4/5 du plateau).

Pour l'affichage nous avons tout d'abord essayé d'utiliser la librairie FSML, mais nous avons rencontré de nombreux problème en particulier un problème de compatibilité entre processeurs tournant sur 32 bits ou sur 64. Après une grande perte de temps nous avons renoncé à cette librairie et pour utiliser « conio » pour ajouter de la couleur à notre univers grâce a la fonction « affichage() ».

Etre vivant :

Nous allons maintenant vous présenter les différents êtres vivant présent dans cet univers et leur particularité. Chaque être vivant est une classe héritant de la classe être vivant.

- Les moutons : possédant une espérance de vie de 48 tours
- Les loups : possédant une espérance de vie de 60 tous
- L'herbe

Les fonctions :

Ci-dessous chaque fonction va être détaillée pour une meilleure compréhension.

La fonction « affichage() » :

Elle permet d'afficher notre tableau avec ce qu'il contient, nous l'avons séparé en deux parties,

- 1^{er} cas le sol n'est pas de l'herbe mais du sable
- 2^{ème} cas le sol est de l'herbe

L'arrière plan est bleu.

Le **sable** est représenté en jaune par la lettre « s ».

Les **moutons** ainsi que les **loups** sont représenté en blanc par les lettres « m » pour les moutons et « l » pour les loups.

L'**herbe** est représenté en vert par la lettre « h ».

Pour ce faire il nous a suffi de tester quel était le type (attribut) de la case sachant que 0 correspond à une case vide, 1 une case contenant du sable, 2 une case contenant de l'herbe, 3 une case contenant un mouton, et 4 un loup en prenant compte du nombre d'être vivant présent sur chaque case.

La fonction « `initialisation()` » :

Cette fonction permet comme son nom l'indique d'initialiser notre tableau.

On commence par créer un désert en initialisant chaque case avec du sable, on ajoute ensuite aléatoirement de l'herbe. Après cela on place aléatoirement le nombre de mouton et de loup demandé précédemment par l'utilisateur sur le plateau en prenant soit de ne pas mettre trop d'être vivant sur la même case.

La fonction « `majSelsMineraux()` » :

Cette fonction gère le nombre de sels minéraux présent sur chaque case en :

- Ajoutant de l'herbe si la case contient du sable plus des sels
- Remplaçant l'herbe par du sable s'il n'y a plus de sels minéraux
- Enlevant un sel pour permettre à l'herbe si elle est présente sur la case de se nourrir.

La fonction « `accouplement()` » :

Cette fonction a pour but de faire se reproduire les moutons ainsi que les loups. Pour cela nous avons séparé la fonction en deux parties, une pour les loups, une pour les moutons.

Lorsque deux animaux de la même espèce se retrouvent sur la même case et qu'ils sont en mesure de se reproduire (tous les 12mois) un mouton voit le jour sur le plateau de manière aléatoire.

La fonction « manger() » :

Cette fonction a pour but de nourrir les animaux. Elle effectue plusieurs opérations :

- S'il y a de l'herbe et un mouton sur la case, le mouton mange, l'herbe réapparaît immédiatement s'il y a des sels minéraux.
- Si le mouton ne peut manger d'herbe, il peut mourir si l'attribut « faim » arrive à 0, sinon cette variable perd un point.
- S'il y a un mouton et un loup sur la case, le loup mange le mouton et des sels minéraux apparaissent aux alentours.
- S'il n'y a pas de mouton mais un loup, celui-ci peut mourir de faim de la même manière que le mouton.

La fonction « age() » :

Cette fonction sert à savoir si le mouton ou le loup meurt de vieillesse ou s'ils restent en vie. A chaque tour son espérance de vie diminue de 1 jusqu'à ce qu'il atteigne 0. Lorsqu'un animal meurt il est supprimé du plateau.

La fonction « tuerie() » :

Cette fonction est utilisée lorsqu'un mouton se fait manger par un loup pour répandre des sels minéraux tout autour de son lieu de décès.

La fonction « deces() » :

Cette fonction est complémentaire à la fonction « tuerie() » en effet lorsqu'un animal meurt de vieillesse ou de faim il répand des sels minéraux tout autour de lui.

La fonction « déplacement() » :

Cette fonction permet aux animaux de se déplacer sur la plateau de deux cases par tour dans n'importe quelle direction aléatoirement. Si l'animal veut sortir du tableau il se retrouve de l'autre côté (haut -> bas, droite -> gauche ect..)

Le déroulement

Après avoir initialiser les différentes variables et demander à l'utilisateur de choisir les paramètres de la partie le déroulement commence.

Tout d'abord les animaux se déplacent grâce à la fonction « déplacement() » puis la fonction « age() » augmente l'âge des être vivants d'un mois ou les fait mourir. Ensuite les animaux toujours en vie mangeront s'ils le peuvent ou mourront via la fonction « manger() ». Après avoir mangé les survivants s'accoupleront si deux animaux de la même espèce et étant dans la bonne période se retrouvent sur la même case avec la fonction « accouplement() ». Pour finir on ajoute ou on diminue le nombre de sels minéraux et d'herbe sur le plateau par l'intermédiaire de la fonction « majSelsMineraux() » et on affiche avec la fonction « affichage() ». On répète ces opérations tant que la limite de temps n'est pas atteinte.

Conclusion :

Grâce à ce projet nous avons pu mettre en application le nouveau langage appris, à savoir le C++. Nous avons trouvé celui-ci très agréable et plus simple d'utilisation que ceux connus jusqu'à présent bien que les erreurs peuvent être tout aussi difficiles à repérer si ce n'est plus. Le thème abordé par le sujet était à la fois intéressant et amusant, ce qui a augmenté notre motivation et notre investissement dans le projet. De plus l'utilisation des classes simplifie bon nombre de problèmes d'ordre algorithmique et simplifie la réalisation de code parfois complexe. Ce projet nous a donc donné envie de découvrir d'autres langages et voir s'ils en existent des complètement différents au niveau de la conception (C par rapport au C++) comme de l'utilisation.

Annexe :

Diagramme des classes

