Partie MASON

# Classe Animal

La classe *Animal* est une classe abstraite définissant les comportements communs à tous les animaux.

## Comportement périodique "step"

DEBUT:

Appeler la fonction abstraite Action();

Incrémenter l’âge;

Diminuer le poids;

SI (poids < POIDS\_MIN

OU

age > AGE\_MAX) ALORS:

Mourir;

FINSI

FIN

La classe *Animal* définit également les fonctions suivantes:

* **CanBeEaten()** qui détermine si l'animal peut être mangé.
* **Attacked()** qui détermine le comportement à adopter en cas d'attaque (fuir ou se cacher).
* **Flee()** permet à l'animal de s'enfuir loin de son prédateur.
* **Hide()** permet à l'animal de se cacher.

# Classe Herbivore

La classe Herbivore est héritée de la classe Animal et définit des comportements spécifiques liés à la végétation.

## Comportement périodique "Action"

DEBUT:

SI poids < POIDS\_MIN + Jours\_de\_jeune\*Poids\_consomé\_par\_\_jour ALORS:

Appeler Chasser();

SINON:

Se déplacer au hasard;

FINSI

FIN

## Fonction "chasser"

DEBUT:

PercevoirNourriture();

Nourriture = NourritureLaPlusInteressante();

SI Nourriture est sur la case courante ALORS:

Manger();

SINON:

SeDéplacerVers(Nourriture);

FINSI

FIN

## Fonction "percevoirNourriture"

DEBUT:

POUR chaque case *C* dans un rayon *perception* de moi:

*maPerception*[Xrel][Yrel].x = C.x;

*maPerception*[Xrel][Yrel].y = C.y;

*maPerception*[Xrel][Yrel].qte= C.qte\_Vegetation;

FINPOUR

FIN

## Fonction "NourritureLaPlusInteressante"

DEBUT:

Liste Cases\_OK;

Int Dist\_min

POUR chaque case *C* de *maPerception*:

Si *C*.qte > besoin\_Quot/2 ALORS:

SI distanceÀ(*C*) < Dist\_min ALORS:

vider Cases\_OK;

Dist-min = distanceÀ(*C*);

ajouter *C* à Cases\_OK;

SINON SI distanceÀ(*C*)=Dist\_min ALORS:

Ajouter *C* à Cases\_OK;

FINSI

FINSI

FINPOUR

Retourner une case aléatoire de Cases\_OK;

FIN:

# Classe Carnivore

# Classe AnimalFactory

# Classe VegetationManager

En premier lieu, nous avions pensé créer la végétation comme une entité à part entière. Cette entité *steppable* devait entrer en interaction avec les animaux herbivore, pour notamment être perçue ou mangée. Cependant, la nécessité de créer une entité par case (soit 100 000 entités pour une carte de 100x100 cases) nous avons rencontré d'importants soucis de performance. C'est pourquoi nous avons opté pour un objet unique, le *VegetationManager*, en charge d'une matrice d'entiers symbolisant la végétation sur les différentes cases de la carte.

La classe *VegetationManager* est donc responsable de l'initialisation de la matrice de végétation de la carte et lui attribue aléatoirement une hauteur initiale aléatoire de végétation. *VegetationManager* est également charge de rendre compte de la croissance de cette végétation à chaque tour en mettant à jour la matrice de végétation ainsi que sa représentation graphique.

# Classe StatsManager

La classe *StatsManager* a pour fonction de conserver les différentes données de simulation et d'en calculer les statistiques. En effet, les différentes actions entreprises par les animaux (manger, naître, mourir etc.) sont entreposées au sein de *StatsManager* qui en déduit alors les taux de natalité, les taux de mortalité, les populations, leur âge et poids moyen etc.

*StatsManager* est aussi en charge de la création et de la mise à jour des graphes tracés au cours de la simulation.

Partie Amélioration possible MASON

Equilibrage des espèces

Espèces marines

Espèces omnivores

Reproduction

Gestion des cycles jour/nuit

Génération d'évènement épidémies/Catastrophes naturelles

Croissance aléatoire de la végétation

Création de milieux différents (déserts, forêt)

Editeur de cartes

Comportements de meutes