Rendu 2 : Camille Venisse (375454) et Edgar Ruault (376265)

Hiérarchie de classe des entités de Lifeform

SUPERCLASSE LIFEFORM

La classe Lifeform est une superclasse dont vont hériter les classes Algue, Corail et Scavenger. Elle permet ainsi de mettre en œuvre le principe de réutilisation en fournissant des méthodes communes aux classes des différentes entités. La méthode `save()` est virtuelle car on a besoin de la redéfinir dans les sous-classes tout en utilisant sa définition originale qui permet de sauvegarder l'âge et la position seulement.

Attributs :

- `S2d position` : position d'une entité (algue, corail ou scavenger) dans le monde
- `double age` : âge de l'entité (algue, corail ou scavenger)
- `double success` : booléen qui permet au module simulation de savoir s'il y a eu une erreur lors de la lecture des données des différentes entités

Méthodes :

- Constructeur Lifeform
- Getters pour accéder à l'âge et la position des entités ainsi qu'au booléen de succès de lecture
- `void incrementer()` : méthode pour incrémenter l'âge d'une entité (ici, uniquement l'âge d'une algue pour le rendu2)
- `virtual void save(fstream &fichier_sauvegarde)` : ajoute la position et l'âge à fichier_sauvegarde, cette méthode sera réutilisée et ajustée dans les sous-classes

CLASSE ALGUE : hérite de Lifeform

Méthode :

- `void draw()` : dessine l'algue

CLASSE CORAIL : hérite de Lifeform

Attributs : Ces attributs nous permettent d'accéder en tout temps aux informations relatives à un corail particulier (qui est une instance de la classe Corail) comme par exemple ses segments, stockés sous forme d'un vecteur d'instances de la classe Segment (du module shape), mettant ainsi en oeuvre le principe d'abstraction.

- `S2d end :` coordonnées de l'extrémité de l'effecteur du corail
- `bool statut_cor` : état mort (0) ou vivant (1) du corail
- `bool dir_rot` : direction de rotation de l'effecteur du corail (sens TRIGO ou INVTRIGO)
- `bool statut_dev` : état EXTEND ou REPRO
- `unsigned nbseg` : nombre de segments du corail
- `vector<Segment> segments` : vecteur qui contient tous les segments du corail

Méthodes :

- `void addSegment(Segment seg, bool reading)` : fait les vérifications sur le segment courant et l'ajoute au vecteur de segments du corail
- `void Superposition()` : vérifier qu'il n'y ait pas de superposition entre segments

Rendu 2 : Camille Venisse (375454) et Edgar Ruault (376265)

- Getters pour accéder à l'id, le statut, la direction de rotation, le statut de développement, le nombre de segments, l'extrémité de l'effecteur et les segments du corail
- `void draw()` : dessine le corail
- `void save(fstream &fichier_sauvegarde)` : ajoute au fichier de sauvegarde les informations des coraux

CLASSE SCAVENGER :

Attributs :

- `double rayon` : rayon du scavenger
- `bool statut` : statut MANGE ou LIBRE
- `unsigned corail_id_cible` : identifiant du corail ciblé par le scavenger
- `void draw()` : dessine le scavenger
- `void save(fstream &fichier_sauvegarde)` : ajoute au fichier de sauvegarde les informations des scavengers

Méthodes :

Getters pour le rayon, statut, id du corail cible,

Structuration des données des autres entités du Modèle

Attributs de Simulation :

- `vector<Corail> coraux` : vecteur qui contient toutes les instances de la classe Corail, c'est-à-dire tous les coraux de la simulation courante
- `vector<Algue> algues` : vecteur qui contient toutes les instances de la classe Algue, c'est-à-dire toutes les algues de la simulation courante
- `vector<Scavenger> scavengers` : vecteur qui contient toutes les instances de la classe Scavenger, c'est-à-dire tous les scavengers de la simulation courante
- `bool sim_success` : échec ou succès de la lecture de fichier
- `default_random_engine generator:` permet de générer des positions aléatoires pour la naissance des algues

Ensembles gérés par Simulation :

Les ensembles de coraux, algues et scavengers sont mémorisés dans des vecteurs qui contiennent toutes les entités du monde sous forme d'instances de leur classe respective (Algue, Corail, Scavenger du module lifeform (principe d'abstraction). Ces vecteurs sont remplis au moment de l'appel de la méthode `readFile`. On pourra ainsi accéder à tout moment à toutes les données d'une entité donnée, par exemple dans la fonction de sauvegarde qui parcoure les différents vecteurs pour écrire dans le fichier de sauvegarde leurs données.

Types mis en oeuvre dans le module shape :

Le module shape contient une structure S2d qui permet de définir des points/vecteurs du plan et une classe Segment caractérisée par son angle et sa longueur. Pour le dessin des différentes formes, ce sont des fonctions spécifiques : `draw_line()`, `draw_circle()`, `draw_square()` qui sont en dehors de toute classe/structure (principe d'abstraction) et qui font un appel direct aux fonctions de graphic.