Fichier des réponses

Exercice P1 : Vecteur 3D

Question P1.1 : Comment représentez-vous ces vecteurs ? Comment sont-ils organisés : quels attributs ? Quelles méthodes ? Quels droits d'accès ?

Les vecteurs sont représentés par des tableaux statiques de 3 nombres réels (3 dimensions). Ils ont un seul attribut privé qui est un array contenant les coordonnées, dans l'ordre x,y,z.

Les méthodes, toutes publiques, sont les opérateurs conventionnels mathématiques sur les vecteurs dans l'espace, ainsi que des méthodes pour accéder et modifier les coordonnées, ainsi que les constructeurs pour les initialiser (par défaut, le vecteur nul, car c'est l'élément neutre pour l'addition/la soustraction).

Concernant la méthode pour modifier les coordonnées (set_coord), deux variantes existent (par surcharge), soit une pour modifier l'ensemble des coordonnées et l'autre pour ne modifier qu'un élément. Cela permet de ne pouvoir changer qu'une seule coordonnée et d'éviter la copie lors que le vecteur change selon un axe, mais d'éviter la redondance en appelant 3 fois la fonction si les trois coordonnées changent.

Exercice P4 : Révision des vecteurs

Question P4.1 : Avez-vous ajouté un constructeur de copie ? Pourquoi (justifier votre choix) ?

On a décidé de garder le constructeur de copie par défaut, car il peut être utile pour éviter de devoir utiliser le constructeur normal et accéder à chaque coordonnée x, y, z individuellement, comme par exemple si l'on désire faire des copies de vitesses.

Question P4.2 : Si l'on souhaitait ajouter un constructeur par coordonnées sphériques (deux angles et une longueur)

a) *Que cela impliquerait-il au niveau des attributs de la classe ?*

   Soit on peut garder les attributs comme ils sont mais on devra utiliser des sinusoïdes pour passer de coordonnées sphériques en cartésiennes.
   Soit on peut ajouter un nouveau tableau de trois éléments (deux angles et une longueur) et lors de la construction on complète les deux tableaux pour contenir la même position.

b) *Quelle serait la difficulté majeure (voire l'impossibilité) de sa réalisation en C++ ? (C'est d'ailleurs pour cela qu'on ne vous demande pas de faire un tel constructeur)*

   Les constructeurs en cartésien et en sphérique utiliseraient les mêmes arguments (trois nombres réelles), le compilateur ne pourra pas les distinguer ce qui cause des erreurs. Il faudrait ajouter un quatrième argument pour les distinguer les constructeurs de mêmes que toutes les méthodes.

Remarque : L'utilisation des coordonnées sphériques n'est tout simplement pas pratique, car pour l'addition de deux vecteurs, il faudrait mettre le tout en coordonnées cartésiennes, sommer, puis remettre en sphériques. Cela est long, inutile et compliqué.

Question P4.3 : Quels opérateurs avez-vous introduits ?

- Addition (+ et +=)
- Soustraction (- et -=) ainsi que l'opposé (-)
- Multiplication par scalaire (*, toujours de la forme scalaire * vecteur, par choix) (division : 1 / scalaire * vecteur)
- Produit scalaire/vectoriel (*/^)
- Comparaison (== / !=)
- Affichage << (pour les tests/débugs)

Exercice P5 : Masses

Question P5.1 : Comment avez-vous implémenté l'ensemble de ressorts ?

Puisque les ressorts sont partagés, on utilise des tableaux dynamiques de pointeurs. Cela permet de pouvoir modifier facilement un ressort donné, et dynamiques car la taille peut varier tout au long de la simulation.

Exercice P7 : Intégrateurs

Question P7.1 : Comment représentez-vous la classe Integrateur ? Expliquer votre conception (attributs, interface…)

La classe Integrateur ne contient aucun attribut mais une méthode évolue qui modifie les attributs (position, vitesse) d'une masse (passée par référence) pour la représenter plus tard dans le temps. Cette classe est une classe virtuelle pure.

Question 7.2 : Quelle est la relation entre les classes Integrateur et IntegrateurEuler

La classe IntegrateurEuler est une sous-classe de Integrateur. On a donc du polymorphisme pour la méthode évolue.

Remarque : Pour le projet, nous n'avons pas implémenté une classe IntegrateurEuler mais plutôt une classe IntegrateurEulerCromer, qui y ressemble beaucoup.

Exercice P8 : Tissus (version simple)

Question P8.1 : Lesquelles des méthodes précendentes avez-vous implémentées ? Les avez-vous mises en public ou private ? Précisez pour chacune et expliquer pourquoi ?

Le constructeur de Tissu prend un tableau dynamique de masses (reliées à aucun ressort) et initialise le tableau dynamique de ressorts à une liste vide. On a juste un ensemble de masses.

Une fois le Tissu créé avec toutes les masses, on peut ajouter des ressorts un à la fois avec la méthode connect, qui recevra comme attributs les adresses des deux masses auxquelles ils sont reliées, une longueur au repos et une constante de rappel. Cette méthode est publique car on crée des tissus depuis un autre fichier. Chaque fois que l'on fait appel au constructeur de ressort, l'adresse du ressort est stockée dans le tableau dynamique de ressorts des deux masses concernées.
Puisque ce constructeur met déjà les adresses des ressorts dans les masses correspondantes, la

méthode connecte_masses n'est pas nécessaire. Cependant, nous en avons quand même créé une, publique, pour les situations où nous souhaiterions supprimer un ressort, ou qu'une erreur serait apparue.

Nous avons aussi implémenté une méthode check, privée, que nous utilisons pour vérifier après chaque modification de notre tissu que tout c'est bien passé.

Exercice P9 : Système (Dessinable, SupportADessin et Système)

Question P9.1 : En termes de POO, quelle est donc la nature de la méthode dessine ?

C'est une méthode virtuelle, qui ne retourne rien et ne prend rien en paramètre.

Question P9.4 : Comment représentez-vous la classe Système ? Expliquer votre conception (attributs, interface…)

La classe Système contient trois attributs : un tableau dynamique de pointeurs sur des tissus, un tableau dynamique de pointeurs sur des contraintes (même si la classe Contrainte n'existe pas encore) et finalement un pointeur sur un intégrateur.
Elle a trois modificateurs qui permettent par exemple d'ajouter un tissu, d'ajouter une contrainte (vide pour l'instant) ou de changer l'intégrateur.
Elle possède une méthode dessine qui la dessine, c'est-à-dire les tissus.
Il y a encore la méthode evolue ayant comme paramètres un pas de temps et le temps absolu (utilisé pour les contraintes, rajouté par la suite). Cette méthode va dans un premier temps mettre à jour les forces, puis dans un deuxième temps appliquer les contraintes (modifie les forces et les vitesses des masses), puis finalement faire avancer d'un pas de temps tous les tissus.

Exercice P14 : Autres intégrateurs

Question P14.1 : Où cela s'intègre-t-il dans votre projet/conception ? Quels changements cela engendre-t-il ?

Ce nouvel intégrateur vient "remplacer" (on garde l'intégrateur d'Euler-Cromer, on peut utiliser l'un ou l'autre) pour améliorer notre simulation. Aucun changement n'est à faire, les tests étant déjà fait, mais pour la suite (extension) il sera dorénavant possible de choisir entre les deux, voire de changer en cours de route.