RÉPONSES

[Question P1.1]

Comment représentez-vous ces vecteurs ? Comment sont-ils organisés: quels attributs? quelles méthodes? Quels droits d'accès ?

Pour représenter les vecteurs, nous utilisons un tableau dynamique de *double*. De ce fait, nous pourrons utiliser des vecteurs de tailles différentes. Néanmoins, comme nous sommes dans le cadre d'un problème physique, ils se limiteront certainement au plus à une taille trois (ou taille six quand deux vecteurs sont concaténé).

L'utilisateur peut accéder aux fonctions *affiche, augmente* et *compare* dont le rôle est d'afficher les vecteurs, d'ajouter une dimension à un vecteur en lui associant une valeur et de comparer deux vecteurs.

[Question P1.2]

Quel choix avez-vous fait pour les opérations entre vecteurs de dimensions différentes ?

Nous ne voulons pas rendre possible de telles opérations c'est pourquoi, nous avons choisi de gérer ces cas au moyen d'exceptions. Ce procéder est systématique et clair. Il permet aussi, s'il y en a besoin, d'avoir une gestion des erreurs plus développée (niveau d'erreur, code de l'erreur, arrêt du programme,...).

[Question P4.1]

Avez-vous ajouté un constructeur de copie? Pourquoi ?

Nous avons décidé d'ajouter un constructeur de copie, afin de pouvoir l'utiliser dans les définitions des surcharges d'opérateurs et aussi parce que c'est un moyen pratique de dupliquer les vecteurs. Cela pourra être utile dans la suite du projet.

[Question P4.2]

Si l'on souhaitait ajouter un constructeur par coordonnées sphériques (deux angles et une longueur) pour les vecteurs de dimension 3,

a] que cela impliquerait-il au niveau des attributs de la classe ?

Il faudrait que ce constructeur convertisse les coordonnées sphériques entrées par l'utilisateur en coordonnées cartésienne avant de les affecter au vecteur.

b] quelle serait la difficulté majeur (voire l'impossibilité) de sa réalisation en C++?

Si la classe avait un autre constructeur prenant trois *double* pour les mettre dans des composantes cartésiennes, il y aurait alors une difficulté d'utilisé le bon constructeur au bon moment, car le constructeur de coordonnées polaires prendrait également trois *double*.

[Question P4.3]

Quels opérateurs avez-vous introduit ?

Compare (==), affiche(<<), addition, soustraction, oppose(-), multiplication par un scalaire(*), multiplication vectorielle(*), produit mixte(^) et ajout(+=)

[Question P5.1]

Comment avez-vous implémenté la masse et la masse volumique des boules : comme attribut ou comme méthode ?

Nous avons implémenté la masse volumique comme méthode constante prenant en argument une boule et calculant la masse volumique en fonction de la masse et du rayon. Cela nous permet de garantir de ne pas avoir d'incohérences entre la masse le rayon et la masse volumique.

[Question P5.2]

Comment avez-vous implémenté les positions, angles, vitesse et vitesse angulaire des boules : comme attributs ou comme méthodes ?

Les informations concernant le mouvement des boules sont contenu dans les attributs *omega* et *omegaP* avec la convention que les variables du mouvement figure dans le tableau dans l'ordre suivant: x,y,z, α, β,γ.

[Question P6.1]

Comment représentez-vous la classe *Integrateur*?

La classe *Integrateur* n'a pas d'attributs propres, car elle est plus une instance-outil, qu'une instance objet. C'est à dire qu'elle va permettre de calculer l'évolution des boules en fonction du temps.

[Question P6.2]

Quelle est la relation entre les classes *Integrateur* et *IntegrateurEuler* ?

Les classes *Integrateur* et *IntegrateurEuler* sont reliés par un lien d'héritage. *IntegrateurEuler* est un *Integrateur*. *Integrateur* étant destiné à être spécialisé en une certaine méthode d'intégration, la méthode qu'il contient est de type virtuelle pure. Cette méthode (*integre*) sera définie dans les sous-classes de la super-classe *Integrateur* ; la seule sous-classe actuelle étant *IntegrateurEuler*. Dans cette dernière, la méthode *integre* est programmée pour calculer de manière numérique la position et la vitesse des boules au moyen de la méthode de Euler-Cromer.

[Question P7.1] Quelle est la relation entre les classes *Objet* et *Boule* ?

Boule est un *Objet*. Il y a donc une relation d'héritage entre *Boule* et *Objet*. Actuellement cela ne concerne que *Boule*, mais viendront ensuite les parois et le sol.

[Question P8.1]

En termes de POO, quelle est donc la nature de la méthode dessine ?

Tout *Objet* est un dessinable et donc hérite de *Dessinable*. *Dessinable* a une méthode *dessine* qui est virtuelle pure et qui force donc la redéfinition dans les sous classe. *Dessine* est donc une méthode virtuelle pure (dans *Dessinable*) qui est redéfinie (ou substituée) dans tous les dessinables de manière à afficher chaque *Objet* d'une façon qui lui est propre.

[Question P8.2]

Quelle est la bonne façon de le faire dans un cadre de programmation orientée-objet ?

Il est bon d'utiliser le polymorphisme au moyen de méthode *virtual* dans la superclasse afin que chaque sous-classe puisse mettre en œuvre sa méthode spécifique.

Une bonne manutention de ces différentes classes pourra se faire au moyen d'un tableau hétérogène d*'objets*.

[Question P8.3]

A quoi faut-il faire attention pour les classes contenant des pointeurs ? Quelles solutions peut-on envisager ?

Lors de l'utilisation de pointeurs, il faut être vigilant à l'allocation et à la libération des emplacements mémoires. Ce problème peut être allégé par l'utilisation de pointeurs intelligents.

Ces derniers empêchent également à deux pointeurs de pointer sur la même instance, ainsi ils assurent une vie indépendante à chaque pointeur (i.e. on ne se retrouve pas avec un pointeur qui pointe sur rien). Néanmoins, dans le projet, comme les mêmes boules se trouvent dans deux tableaux différents, ce n'est pas l'effet désiré. On utilisera donc des pointeurs à la C.

[Question P8.4]

Comment représentez-vous la classe Systeme ? Expliquez votre conception (attributs, interface, ...).

En attribut, la classe *Systeme* est constituée d'une collection hétérogène implémentée sous la forme d'un tableau dynamique de pointeurs sur des *Objets* et d'un tableau formé uniquement des boules du tableau *d'Objets*.

Pour ce qui est des méthodes,...