TP de C++ n° 5

Simulation de mobiles (seconde partie)

Ce TP fait suite au précédent. Il vous est demandé de façon générale de bien structurer le code, avec pour chaque classe un fichier `.h` *et un fichier* `.cpp`*, et votre fonction* `main` *à part faisant appel aux différents tests. Vous devez rendre les deux TP ensemble sous la forme d'une archive qui, une fois extraite, doit révéler l'arborescence suivante :*

où aucun fichier ne doit comporter de fonction `main`*, où le fichier* `tests.cpp` *doit contenir tous les tests demandés (y compris ceux du TP n° 4), et où le fichier* `.pdf` *doit être un compte-rendu au format PDF présentant le travail réalisé et incluant des indications de conception, d'implémentation, et une justification des tests vous permettant d'affirmer que le code ne présente aucun défaut. Cette archive doit être déposée au plus tard le* samedi 03 novembre 2018 à 18h00 *sur l'Espace Numérique de Travail dédié au cours.*

— : —

Le but de ce TP est d'utiliser l'infrastructure d'un logiciel de simulation de mobiles en trois dimensions du TP précédent pour coder une simulation de satellites. De façon générale, il vous est demandé :

- d'utiliser le passage des arguments par référence, selon ce qui a été indiqué en cours ;
- de faire attention aux fuites de mémoire (un `delete` pour un `new`) ;
- utilsier le mot clef `const` chaque fois que c'est utile, pour les arguments des méthodes comme pour les méthodes elles-mêmes ;
- de ne pas mettre d'attributs publics (sauf dans la classe `Vecteur3D`), à moins d'avoir une justification pour cela.

— : —

Exercice 1 : *Vecteurs et opérateurs*

La classe `Vecteur3D` ne respecte pas un des principes de l'approche orientée-objet : l'*encapsulation*. On va modifier cette classe pour la rendre à la fois plus sûre et plus pratique à utiliser.

1°) Modifier la classe pour que ses attributs soient `private` et ajouter un opérateur `[]` prenant un entier (`int`) entre `0` et `2` comme argument et retournant les coordonnées `x`, `y` ou `z`.

2°) Ajouter un opérateur d'addition entre deux vecteurs, un opérateur de multiplication entre un double et un vecteur. Plus généralement, ajouter tous les opérateurs que vous jugerez utiles («, +=, =, ==, ...) afin de simplifier les écritures dans les classes `Mobile` et `MobilePesant`.

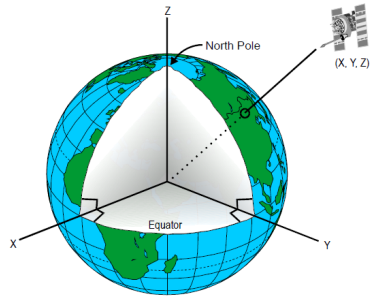
3°) Modifier en conséquence le code des classes Mobile et MobilePesant du TP n° 4[1]. Écrire des tests de la classe `Vecteur3D` dans une fonction `bool testVecteur3D()`.

Exercice 2 : *La Terre est ronde, en fait...*

Les modèles de Terre et de gravité du TP précédent sont trop simplistes. D'abord, la Terre n'est pas plate, mais ronde[2]. Ensuite, la gravité n'est pas constante : elle diminue quand on s'éloigne de la Terre.

1. Je sais, ce n'est pas le truc le plus passionnant que vous aurez fait de votre vie.
2. En fait, elle est même plutôt ellipsoïdale, voire carrément bosselée.

On va donc changer de modèle de coordonnées :



Les coordonnées `x`, `y`, `z` sont maintenant par rapport au centre de la Terre :

— l'axe X : passe par l'intersection de l'équateur et du méridien de Greenwich ;
— l'axe Y : est dans le plan de l'équateur, orthogonal au précédent ;
— l'axe Z : joint le pôle Sud au pôle Nord.

Par exemple, $\vec{M} = \begin{vmatrix} 6378000 \\ 0 \\ 0 \end{vmatrix}$ est situé à la surface de la Terre, à l'intersection de l'équateur et du méridien de Greenwich.

Pour cela, on va créer une classe Terre, qui sera caractérisée par :

— une constante `GM` $= 3,986.10^{14}$ m^3.s^{-2} ;
— un rayon `RT` $= 6378000$ m.

Dans un repère tel que celui présenté sur la figure, la gravité en un point $\vec{P}$ vaut : $\vec{g} = -\dfrac{\texttt{GM}}{\|\vec{P}\|^3}\vec{P}$.

1°) Écrire une méthode intitulée `gravite` qui calcule la gravité (`Vecteur3D`).

2°) La Terre est non seulement ronde, mais aussi unique ! Mettez en œuvre le motif de conception (*design pattern*) « Singleton », afin d'avoir disponible à tout instant une seule et unique instance de Terre. Utiliser la description et une des méthodes qui se trouvent dans le polycopié du cours.

3°) Écrire des tests de la classe `Terre` dans une fonction `bool testTerre()`.

4°) Maintenant que la Terre est prête, on va l'utiliser dans la classe MobilePesant. Remplacer la valeur constante jusqu'à présent, à savoir $\vec{g} = (0\ 0\ -9,81)$, par un appel au modèle de gravité de la Terre (singleton).

Exercice 3 : ***Simulation de satellite***

On va faire un test grandeur nature.

1°) Réalisez une petite simulation (dans une fonction `bool testSatellite1()`) avec un satellite à $h = 200$ km d'altitude, au dessus de l'équateur, un vecteur vitesse orthogonal au vecteur position $\vec{M}$ et de module $\sqrt{\frac{\texttt{GM}}{\texttt{RT}+h}}$ (cela devrait faire environ 7,8 km.s^{-1}).

2°) Toujours dans `bool testSatellite1()`, exécuter la simulation durant $T = 2\pi\sqrt{\frac{(\texttt{RT}+h)^3}{\texttt{GM}}}$ s (cela devrait faire environ 1 h 29 min) avec un pas de temps de 0,1 s. Votre satellite devrait en principe avoir fait le tour de la Terre et être revenu à peu près à sa position de départ.

3°) Pour vérifier cela, ajouter tout le nécessaire dans les classes `Terre`, `Vecteur3D` et `Mobile` pour calculer la distance entre deux points. Vérifier dans `bool testSatellite1()` que la distance entre le point de départ et le point d'arrivée (au bout du temps T) de votre satellite est de quelques kilomètres, voire quelques mètres.

4°) Le module de la vitesse d'un satellite à une altitude h est $\sqrt{\frac{\texttt{GM}}{\texttt{RT}+h}}$. Créer un constructeur pour votre classe de mobile pesant qui prenne en argument quatre `double` — dans l'ordre, ses coordonnées X, Y, Z et son *inclinaison* : il s'agit de l'angle entre le plan de l'équateur (plan formé par les axes X et Y) et le plan de l'orbite (le plan formé par les vecteurs position et vitesse du mobile : il faut donc que votre vecteur vitesse soit tel que le plan formé par lui et le vecteur position fasse le bon angle avec le plan de l'équateur, et qu'il soit orthogonal au vecteur position, et que son module soit égal à celui donné par la formule ci-dessus [3]. Utiliser ce constructeur pour construire le satellite d'observation de la Terre SPOT, que l'on placera au dessus de l'intersection de l'équateur et du méridien de Greewich à une altitude de 820 km avec une inclinaison de 98.7°. Vérifier dans une fonction `testSatellite2()` par un test analogue au précédent que l'on retrouve une période d'environ 1 h 41 min 24 s.

5°) Rechercher sur internet les éléments vous permettant de créer un mobile pesant représentant la Lune, et vérifier que son mouvement est bien conforme à celui attendu.

3. N'hésitez pas à faire un dessin pour mieux comprendre.