Conception

Vecteur3D, Constantes : *Ce sont des "outils" qui servent de fondation pour le reste du projet* :

Vecteur3D : Vecteur3D est une classe qui nous permet de représenter l'espace (R^3). Un objet de la classe, soit un vecteur de R^3, a comme attribut un array de double qui représente ses coordonnées (x,y,z) dans l'espace orthonormé. On définit un ensemble de méthodes qui nous permettent d'effectuer les opérations usuelles qui se font sur des vecteurs de R^3 (en tant que R-espace vectoriel), des getters qui nous aident pour le reste du projet (ceux qui nous paraissent indispensables) et les méthodes de surcharge de l'opérateur d'affichage.

constantes : Constantes est un fichier qui nous permet de retenir et de réutiliser deux constantes essentielles pour le reste du projet : la constante g (champ vectoriel de la pesanteur terrestre), et la constante margin_ qui nous permet de comparer des doubles (la marge d'erreur des approximations).

//==//

SupportADessin : C'est une super classe abstraite qui nous permet de définir sous quelle forme on veut afficher la simulation : soit en mode texte ou en mode graphique.

TextViewer : Cette classe est une sous-classe de SupportADessin. Elle indique comment dessiner les objets dessinables en version text. Elle fait cela en définissant les méthodes virtuelles pures définies dans SupportADessin.

vue_opengl : Cette classe est également une sous-classe de SupportADessin. Elle indique comment dessiner les objets dessinables en version graphique. Son fonctionnement, bien qu'un peu plus complexe que TextViewer (due entre autres à la gestion de la vue et de la position de la caméra), se base sur le même principe que TextViewer.

//==//

Dessinable : C'est une super classe virtuelle qui permet grâce à sa méthode dessine_sur() un double polymorphisme sur à la fois : l'objet à dessiner (l'instance d'une sous classe de dessinable) et le SupportADessin (text ou graphique).

Masse : Cette classe définit l'une des deux composantes qui vont nous permettre de représenter un tissu : des masses ponctuelles. Ses attributs sont donc une masse (en kg), trois vecteurs décrivant sa position, sa vitesse et la résultante des forces qui s'appliquent sur elle, un coefficient de frottement fluide et une collection de

ressorts qui lui sont accrochés. On souhaite dessiner les masses quand on dessinera nos tissus, ce pourquoi la classe masse hérite de Dessinable.

Ressort : Cette classe représente la deuxième composante des tissus. Un ressort lie deux masses et exerce de forces sur ces dernières. Un ressort possède une raideur, une longueur à vide et référence les deux masses qu'elle lie. Idem, la classe Ressort hérite de Dessinable car on souhaite les dessiner . (Cependant on ne souhaite pas dessiner les ressorts textuellement : la méthode dessine(Ressort const&) de TextViewer ne fait rien)

Tissu : Un tissu est représenté par un ensemble de masses et l'ensemble des ressorts qui les lient. Un tissu est dessinable, le dessiner revient à dessiner ses composantes.

Tissu Chaine, Tissu Rectangle, Tissu Disque : Ce sont des sous classes de Tissu. Elles implémentent de nouveaux constructeurs permettant de créer facilement des tissus de la forme souhaitée.

Tissu Composé : C'est aussi une sous classe de Tissu. Cette dernière permet de créer de nouveaux tissus à partir d'anciens tissus en les raccommodant.

Système : Un Système représente un ensemble de tissus et de contraintes à un temps t. Le temps permet d'appliquer les contraintes. Un système est dessinable, le dessiner revient à dessiner son ensemble de tissus soumis à son ensemble de contraintes.

//==//

Intégrateur : Les Intégrateurs sont des classes destinées à faire évoluer le système. Chaque sous classe d'Intégrateur possède une définition différente de la méthode virtuelle pure publique evolue() permettant de faire se mouvoir une masse. La définition varie selon la méthode d'intégration des équations du mouvement utilisée. Chaque sous classe d'Intégrateur a aussi des attributs privés (et en conséquence des Constructeurs) propres à la méthode d'intégration qu'elle implémente.

IntégrateurEulerCromer : Sans attributs supplémentaires, cette méthode d'intégration calcule directement les nouvelles positions et vitesses des masses.

IntégrateurNewmark : Cette méthode d'intégration consiste en une boucle de calculs itératifs. Cela nécessite quatre attributs supplémentaires : trois Vecteur3Ds pour retenir des étapes de calcul intermédiaires et un paramètre (représenté par un double) pour contrôler la convergence de sa boucle de calculs.

IntégrateurRungeKutta :

//==========//

Contrainte : Contrainte est une super-classe qui modélise les actions externes aux tissus modélisés et qui ont un impact sur celui-ci et donc sur son intégration. On les modélise par des boules : un rayon (double) et un centre (Vecteur3D). La classe contrainte est une super classe abstraite qui nous permet de définir plusieurs sous classes qui sont des contraintes "effectives" donc des sous classes instanciables.

Crochet : Crochet est une sous classe de Contrainte. Comme son nom l'indique elle modélise une contrainte qui "accroche" les masses et donc les fige. Pour ce faire, la méthode appliquer() (qui est la redéfinition de la méthode virtuelle pure de Contrainte) met les forces et la vitesse de la masse à 0. Les attributs d'un crochet sont les mêmes que ceux d'une contrainte générale (même constructeur).

Impulsion : Cette deuxième sous classe de Contrainte modélise une force externe qui s'applique sur les tissus. Elle possède en plus des caractéristiques d'une contrainte lambda (une boule), une force (celle de la contrainte appliquée - modélisée par un Vecteur3D -), un temps de début et un temps de fin (des double).

Impulsion_sin : Pour finir, Impulsion_sin est une sous de classe de la classe Impulsion. Elle définit un type particulier d'impulsion : une dont la force exercée évolue dans le temps selon une variation sinusoïdale. Elle est donc caractérisée de la même façon qu'une impulsion, sauf que la force indiquée est une force de référence (celle qui s'applique initialement) ; mais elle est également caractérisée par une fréquence (double) et des tissus cibles. Contrairement aux autres containtes, une impulsion_sin s'applique à des masses cibles définies à leur construction par les attributs : boule et tissus cibles.

Résumé de la conception :