OpenGL - TP 5 et 6

Animation physique

Dans ce double TP, nous allons implémenter deux modèles d'animation physique : les systèmes masse-ressort et les systèmes de collision. Le mini-cours sur les systèmes dynamiques (cf : *TP5-6-Cours.pdf*) vous aidera à comprendre le code fourni et à compléter ce TP. Lisez ce mini-cours.

1 Code fourni

Compilez et exécutez le code fourni : une boule se translate et une boule est fixe.

Code fourni : Observez tout d'abord le fichier *tp5.cpp* : il comporte les méthodes vues précédemment (gestion de la caméra, du pas de temps, dessin de primitives, ...) et de nouvelles méthodes :

- `setDynamics()` qui initialise un objet de la classe `Dynamics` qui permet d'implémenter des modèles d'animation physique ;
- `drawBalls()` et `drawSprings()` qui permettent l'affichage d'une chaîne articulée ;
- `mouseClick()` et `mouseMotion()` qui déterminent les actions contrôlées par la souris. En effet, la position de la boule fixe est contôlée par le bouton gauche de la souris.

Scène initiale : La scène initiale est donc composée de :

- un plan fixe ;
- une boule fixe contrôlée par la souris ;
- une boule avec une vitesse initiale.

Toute l'animation est contrôlée par l'intermédiaire de l'objet `dynamics` de classe `Dynamics`. A chaque pas de temps, la fonction d'animation de `dynamics` est appelée (cf : `updateAnimationParameters()`). Cette fonction va positionner les masses. On va ensuite dessiner les boules représentant les masses ainsi que les ressorts grâce à `drawBalls()` et `drawSprings()`.

2 Schéma d'intégration

Question 1 : Examinez dans le détail les fichiers `dynamics.h` et `dynamics.c`. En particulier, étudiez le calcul des forces et le schéma d'intégration utilisé.
La pesanteur y est prise en compte mais la vitesse de la boule reste constante malgré tout. Pour corriger cela, complétez le schéma d'intégration en mettant à jour la vitesse en fonction des accélérations (stockées dans `forces`). Observez le résultat à l'éxécution.

Question 2 : La boule fixe est un objet dynamique au même titre que la boule mobile. En analysant le programme, expliquez pourquoi elle est fixe.

3 Modèle masse-ressort

Nous allons maintenant construire progressivement une chaîne articulée de boules comme un système masse-ressort. Cela signifie que les boules seront liées deux à deux par un ressort amorti.

Question 3 : Dans la méthode `setDynamics()`, ajoutez un ressort entre la boule fixe et la boule mobile à l'aide de la méthode `addSpring()` de `Dynamics`. Utilisez les variables fournies pour les paramètres du ressort. Observez les résultats sur l'animation en déplaçant la boule fixe à l'aide de la souris.

Question 4 : Modélisez une chaîne de boules accrochée en une extrémité à la boule fixe. Choisissez les mêmes masses et rayons pour toutes les boules et les mêmes paramètres pour tous les ressorts. Paramétrez le nombre de boules créées grâce à la variable `nbBalls`. Observez le résultat.
Modifiez les différents paramètres en observant à chaque fois les réactions du système.

Question 5 : Afin de dissiper les vitesses, modélisez une force d'amortissement visqueux appliquée à chaque boule. Visuellement, quelle est la différence d'effet entre l'amortissement que nous venons d'introduire et l'amortissement des ressorts?

Conseil : Pour bien observer l'influence de la dissipation, augmentez la vitesse initiale des boules et observez l'animation avec et sans amortissement ou faîtes tourner la boule mobile autour de la boule fixe avec et sans amortissement. Pour bien observer l'influence de l'amortissement des ressorts, faîtes varier le coefficient d'amortissement des ressorts.

4 Modèle de collision

Nous allons maintenant gérer les collisions pour obtenir une animation plus réaliste.

Question 6 : Décommentez les lignes nécessaires pour traiter les collisions entre les boules et le sol. Examinez la fonction `collisionBallPlane()` pour comprendre le processus de gestion des collisions. Expérimentez en manipulant la boule fixe.

Question 7 : En vous inspirant très fortement de l'implémentation des collisions boule-plan, traitez la collision boule-boule dans `collisionBallBall()` pour le moment vide.