TP Systèmes masse-ressort

Animation physique

Dans ce TP, nous allons implémenter et expérimenter les systèmes masse-ressort.

1 Code fourni

Remplacez le module `Solver` par celui que vous avez codé. Compilez et exécutez le code fourni : vous devez voir deux masses liées par un ressort.

Code fourni :

- `animPhys.cpp` contient le `main` du programme et gère l'affichage de la scène 3D, l'interface (flèches, `Page-Up` et `Page-Down` pour le déplacement, `Echap` pour quitter, la souris pour le déplacement de la masse 0) ;
- modules `Vec`, `Matrix` et `Solver` vus précédemment ;
- module `Plane` qui définit les propriétés géométriques et mécaniques d'un plan ;
- module `Ball` qui définit les propriétés géométriques et mécaniques d'une boule ;
- module `Spring` qui définit un ressort entre 2 masses `_m1` et `_m2` avec sa raideur `_stiffness`, sa longueur au repos `_length` et son facteur d'amortissement `_damp` ;
- module `Dynamics` qui gère l'animation physique.

Ces deux derniers modules seront complétés pendant le TP.

Scène initiale : La scène initiale est donc composée de :

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

Les 2 masses sont liées par un ressort.

A chaque pas de temps, la fonction d'animation de `Dynamics` est appelée (`animate()`). Cette fonction va positionner les masses. Comme vu en cours, la fonction calcule les forces, met à jour les vitesses et les positions puis traite les collisions.

2 Animation

Pour le moment, il n'y a pas d'animation. Nous allons remédier à cela.

Mise à jour de la position dans l'espace des phases : Complétez la méthode `updatePositions()` de la classe `Dynamics`. Cette méthode utilise les forces déjà calculées pour mettre à jour les positions (stockées dans `positions`) et les vitesses (stockées dans `velocities`). La masse `i` du système masse-ressort est définie par :

- sa masse `masses[i]` dont l'inverse est `invMasses[i]` (l'inverse d'une masse nulle est considéré comme nul : les méthodes `addMass()` et `setMass()` remplissent à la fois `masses` et `invMasses`);
- sa position `positions[i]`;
- sa vitesse `velocities[i]`;
- la somme des forces qui lui sont appliquées `forces[i]`.

Vous pouvez faire appel à votre solveur d'équations différentielles en mettant `_problem` à `P_DYNAMICS` et en lui fournissant la dérivée de votre problème grâce à l'attribut `derivee` de `Solver`.
Une fois cette méthode implémentée, nous obtenons une animation. Nous avons bien une masse fixe et une masse avec une vitesse initiale.

Calcul des forces Cependant, ces masses ne sont soumises à aucune force. Nous allons donc compléter `computeForces()` de `Dynamics` qui calcule, pour chaque masse `i` la somme des forces qui lui sont appliquées et les stocke dans `forces[i]`.

1. Soumettez le système à la gravité en utilisant l'attribut *gravity*. Une des masses ne semblent pas être influencée par la gravité. Pourquoi?
2. Ajoutez les forces dues aux ressorts. Les ressorts sont stockés dans `springs`. La classe `Spring` possède une méthode `addForce(forces, positions, velocities)` qui ajoute à `forces[_m1]` et `forces[_m2]` l'action du ressort amorti. Complétez cette méthode. Etudiez l'influence de la raideur (variable `stiffnessA` dans `initDynamics()` de `Dynamics`).
3. En faisant tourner la masse mobile autour de la masse fixe, il est possible qu'elle tourne indéfiniment. Pourquoi l'amortissement du ressort ne suffit-il pas à résoudre ce problème? Soumettez le système à la viscosité du milieu en utilisant l'attribut `viscosity`. Cela améliore-t-il la situation? Etudiez l'influence du coefficent d'amortissement et la différence entre l'amortissement du milieu et l'amortissement du ressort.

3 Modèles

Nous avons maintenant un système de masse-ressort animé. Nous allons expérimenter différents modèles. Les modèles sont créés dans la méthode `initDynamics()` de `Dynamics`.

1D : Grâce à un paramètre `nbMasses`, créez un modèle de chaîne articulée de `nbMasses` masses fixée par la masse 0. Fixez ensuite l'autre extrémité de la chaîne.

2D : Modélisez un tissu avec 1 puis 2 puis 3 puis 4 points fixes. Observez la nécessité d'ajouter des ressorts pour la rigidité.

3D : Modélisez un cube déformable puis un réseau 3D. Observez la nécessité d'ajouter des ressorts pour la rigidité.