# Systèmes masses ressorts

L'objectif de ce TP est de vous faire modéliser un tissu simulé par un système masses ressorts.

**Contexte.**

**Le simulateur.** Dans le code qui vous est fourni, la classe *Animation::SpringMassSystem* vous permet de décrire un système masse ressort sur lequel des forces peuvent être appliquées. Cette classe possède deux classes internes : Mass (modélisant une masse du système) et Link (modélisant un lien entre deux masses, dans note cas, il s'agira d'un ressort). De plus, cette classe possède plusieurs méthodes importantes vous permettant de paramétrer les forces appliquées sur les masses ainsi que l'intégrateur à utiliser (consulter la documentation pour des informations plus précises) :

- *addForceFunction*. Cette méthode permet d'ajouter une fonction, une lambda fonction ou un foncteur calculant une force à appliquer à une masse. Cette fonction doit posséder la signature suivante :
  *Math::Vector3f (const Mass & mass)*
  La valeur de retour correspond à la force à appliquer à la masse prise en paramètre.
- *addLinkForceFunction*. Cette méthode permet d'ajouter une fonction, une lambda fonction ou un foncteur calculant une force liée à un lien entre deux masses (un ressort dans notre cas). Cette fonction doit posséder la signature suivante :
  *Math ::Vector3f (const Mass & mass1, const Mass & mass2, const Link & link)*
  La valeur de retour correspond à la force à appliquer à la première masse passée en paramètre, la force réciproque sera automatiquement appliquée à la seconde masse.
- *setIntegrator.* Cette méthode de fournir une fonction, une lambda fonction ou un foncteur calculant la nouvelle position et la nouvelle vitesse d'une particule étant données les forces qui lui sont appliquées. Cette fonction doit posséder la signature suivante :
  *::std ::pair<Math ::Vector3f, Math ::Vector3f> (const Mass & previous, const Mass & current, float dt)*
  La valeur de retour est une paire contenant (nouvelle position, nouvelle vitesse).

Les fonctions passées en paramètre de ces trois méthodes sont automatiquement appelées par le moteur de simulation du système masses ressorts. Vous pouvez regarder le code afin de comprendre le fonctionnement de cette classe.

Pour créer un tissu avec un ensemble de masses correctement reliées, la méthode *createPatch* est mise à votre disposition. Par défaut, cette méthode crée un patch dont la première masse (en haut à gauche) est située en (0,0,0) et la masse en bas à droite est située en (tailleX, tailleY, 0). Cette méthode accepte aussi une matrice de transformation pour pouvoir positionner le patch correctement dès sa création.

La mise à jour du système masse ressort s'effectue par l'appel à la méthode *update* qui provoquera le calcul des forces et de l'intégration et générera donc votre animation. Afin de stabiliser la simulation, le système masse ressort possède une fréquence interne de mise à jour (paramétrée par défaut à

1000Hz). Vous pouvez consulter / modifier cette fréquence interne via les méthodes *setInternalPeriod* et *getInternalPeriod*.

**L'affichage.** Pour afficher un système masses ressorts sous la forme d'un tissu, la classe *SceneGraph::Patch* (nœud du graphe de scène) est mise à votre disposition. Cette classe affiche un patch dont la position des sommets peut être changée dynamiquement. Cette classe met à votre disposition la méthode *HelperGl::Buffer<Math::Vector3f> & getVertices()* vous permettant de récupérer le buffer contenant les positions de tous les sommets constituant le patch. Il est à noter que la numérotation des sommets créés pour un patch est la même pour une instance de classe *SceneGraph::Patch* créée par l'appel à la méthode *createPatch* de la classe *Animation::SpringMassSystem*. Pour effectuer l'affichage de votre tissu, il suffit donc de recopier les positions des masses de votre instance de *Animation::SpringMassSystem* dans le buffer retourné par *getVertices*. L'affichage prendra alors automatiquement en compte ces modifications.

**Remarque** : Dans la suite de ce sujet, nous parlerons de fonctions passées en paramètre de méthodes. Ces fonctions peuvent être des fonctions au sens propre, des lambda fonctions ou encore des foncteurs.

**Question 1.** Après avoir créé une nouvelle classe d'application (pensez à intégrer la caméra et la gestion du clavier) :

1. créez une instance de *Animation::SpringMassSystem* et créez le tissu d'un poids de 10Kg, d'une taille 3m x 3m et ayant 50 subdivisions par axe en utilisant la méthode *createPatch*. Le patch créé devra être positionné tel que son centre soit en (0,0,0).
2. créez une instance de *SceneGraph::Patch* qui possédera la même structure que le patch créé au niveau du système masses ressorts (même nombre de subdivisions par axe) et ajoutez cette instance dans un graphe de scène ;
3. dans la méthode *render* de votre application mettez à jour la position des sommets de votre instance de *SceneGraph::Patch* avec la position des masses de votre instance de *Animation::SpringMassSystem* et affichez votre graphe de scène.

Une fois cette structure créée, nous allons pouvoir commencer le TP à proprement parler.

**Question 2.** Programmez une fonction calculant la nouvelle position et vitesse d'une masse en utilisant la méthode d'Euler. La signature de cette fonction devra être compatible avec la méthode *setIntegrator* de la classe *Animation::SpringMassSystem.* Une fois cette fonction programmée, configurez vote instance de système masses ressorts afin qu'elle utilise votre intégrateur.

**Question 3.** Ajoutez au système masses ressorts (via la méthode *addForceFunction*) une fonction calculant la force associée au poids d'une masse en considérant que la gravité vaut $9.81 m.s^{-2}$.

**Question 4.** Ajoutez dans la méthode *render* de votre classe d'application l'appel à la méthode *update* du système masses ressorts. Normalement, votre tissu devrait tomber, à plat et sans jamais s'arrêter.

**Question 5**. En utilisant la méthode *constrainPosition* (méthode qui fixe une masse à sa position courante) de la classe *Animation::SpringMassSystem*, contraignez les positions des masses situées aux extrémités de votre tissu. Relancez l'application. Comme vous pouvez le constater, votre tissu

continue à tomber, à l'exception des masses dont les positions ont été contraintes. Cependant, le tissu n'offre aucune résistance car aucune force n'est appliquée sur les liens entre masses.

**Question 6**. Ajoutez la possibilité, via l'appui sur une touche, de supprimer, une à une, les contraintes de position appliquées aux masses à la question 5. Cela permettra d'effectuer différents tests dans les questions suivantes.

**Question 7.** Ajoutez au système masses ressort (via la méthode *addLinkForceFunction*) une fonction calculant la force associée à un ressort reliant deux masses (sans ajouter d'amortissement pour le moment) et testez le résultat. Normalement, vote système devrait continuellement osciller sans se stabiliser.

**Question 8**. Ajouter une force d'amortissement afin de stabiliser le système en utilisant 0.01 comme coefficient. Normalement, l'animation de vos tissus devrait se stabiliser.

**Question 9.** Via la méthode *addPositionContraint* de la classe *Animation::SpringMassSystem*, ajoutez une contrainte de sol. Ce sol devra être aligné sur le plan (X,Y) à une hauteur de -1.5m. Cette méthode accepte en paramètre une fonction devant avoir la signature suivante : *std ::pair<Math ::Vector3f, Math ::Vector3f> (const Mass & previous, const Mass & current)* Cette fonction retourne la nouvelle position de la masse après application de la contrainte, ainsi que sa nouvelle vitesse. Votre contrainte de sol devra assurer qu'une masse ne puisse pas se trouver sous le sol et devra annuler la composante du vecteur vitesse dirigée vers le sol.

Ajoutez aussi la visualisation de ce sol dans le graphe de scène.

**Question 10**. Via la méthode *addPositionConstraint* de la classe *Animation::SpringMassSystem*, ajoutez une contrainte de type sphère, positionnée en (0,-0.3,-1.0). Ici encore, cette contrainte devra assurer qu'une masse ne pénètre jamais dans la sphère et annulera la composante du vecteur vitesse dirigée vers la sphère.

Ajoutez la visualisation de cette sphère dans le graphe de scène.