Projet de Modélisation et Ingénierie du vivant

Quentin Augrain, Florent Mallard

INSA de Rennes
5INFO

19 janvier 2016

1 Tutoriel

Question 1 : Comme indiqué dans les commentaires de la fonction, il y a deux fonctions dans la boucle :

— Bouger le monde ;

— Afficher le monde.

Cela signifie que la fonction va tout d'abord calculer le prochain mouvement, pour ensuite afficher la nouvelle position. Pour le moment, la phase de calcul n'est pas implémentée et rien ne bouge dans la simulation.

Question 2 : Nous ajoutons la gravité au « force_accumulator », et obtenons le résultat montré à la Figure 1, c'est-à-dire que rien ne se passe. En effet même si nous calculons maintenant les forces, nous ne les appliquons pas au mesh, ce qui résulte en un mesh toujours fixe. Il nous faut donc appliquer ces forces à chaque particule du mesh afin qu'un effet se fasse ressentir.

Question 3 : Maintenant que nous appliquons les forces, le mesh tombe et finit par disparaître de l'écran. Il semble qu'il n'y ait pas de limite physique à sa chute. On peut donc envisager un moyen afin que même si aucune autre interaction que la gravité ne s'applique au mesh, il reste dans le champ de l'écran.

Question 4 : Une autre solution a été proposée, il s'agit de fixer le premier rang des particules. Ainsi, le mesh reste visible et a un effet dit « de rideau ». Pour ce faire, il nous faut nullifier les forces de ces particules. Nous avons compté dix particules sur ce rang, nous avons donc ajouté une boucle comme montré sur la Figure 2. Cette boucle met simplement les forces appliquées à ces dix particules à zéro, empêchant ainsi tout mouvement.

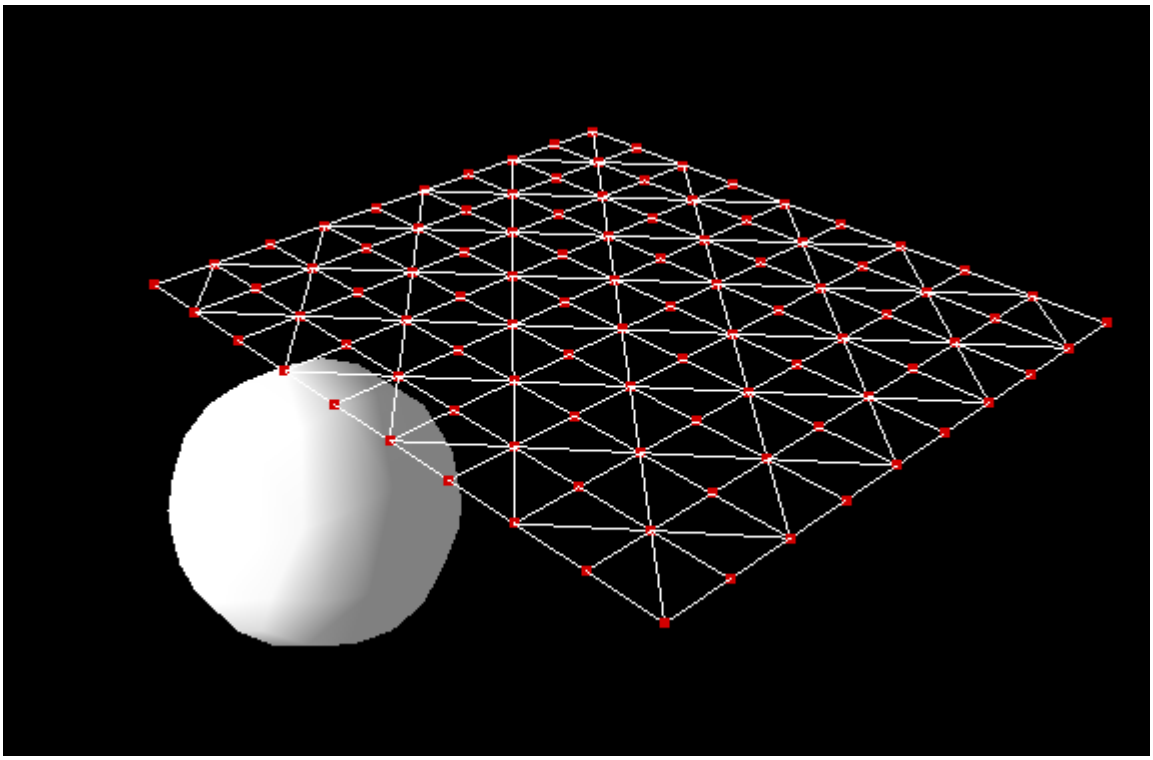

FIGURE 1 – Aucun changement n'est visible même si les forces changent.

```
void Simulator::Update()
{
    // Define the simulation loop (the methods are not in order)
    //ApplyVelocityDamping( ... )
    ComputeForces();

    for (int i = 0; i < 10; i++)
    {
        m_Mesh->particles[i].force_accumulator = Maths::Vector3(0, 0, 0);
    }

    Integrate();
    //UpdateManipulator ( ... )
}
```

FIGURE 2 – Le premier rang reste fixe.

```
for (unsigned int p = 0; p < m_Mesh->particles.size(); p++)
{
    // gravity
    m_Mesh->particles[p].force_accumulator = Maths::Vector3(0, -GRAVITY_CONSTANT, 0);

    // neighbors forces
    Particle *particle = &(m_Mesh->particles[p]);
    for (unsigned int n = 0; n < particle->neighbors.size(); n++)
    {
        Particle *neighbor = particle->neighbors[n];

        float dij = particle->pos.distance(neighbor->pos);
        float dij_init = particle->init_pos.distance(neighbor->init_pos);

        Maths::Vector3 diff_pos = neighbor->pos - particle->pos;
        diff_pos.normalise();

        particle->force_accumulator += diff_pos * K *  (dij - dij_init);
    }
}
```

FIGURE 3 – Mise à jour des voisins pour chaque particule.

Question 5 : Le comportement est pour le moins étrange, car les particules oscillent de plus en plus jusqu'à disparaître. Pour le moment la simulation n'est pas stable, car les forces appliquées aux particules ne se compensent pas. Ceci aboutit, plus ou moins rapidement en fonction des paramètres de simulation choisis, à un écran contenant uniquement le premier rang de particules, demeuré fixe comme précisé à la question précédente. La FIGURE 3 montre notre implémentation.

Question 6 : Le paramètre « dt » définit la vitesse de la simulation à travers le nombre d'itérations à chaque fois. Une valeur plus petite correspond donc à une simulation plus rapide, car on calcule la position et les forces après un instant plus grand, le mouvement a donc été plus important. K définit la rigidité. Celle-ci va aider à obtenir l'effet de rideau désiré. Sachant cela, nous n'avons pas réussi à stabiliser la simulation. En effet même si le rideau est considéré comme plus rigide, les forces des particules ne se compensent pas et donnent le même résultat que précédemment.

Question 7 : Nous ajoutons la boucle demandée qui effectue n appels à « Update » avec un timestep désormais divisé par n. Le but de cette manipulation est d'avoir un mouvement plus petit, car calculé sur un plus petit pas de temps. Grâce à cela les interactions calculées sont plus fines et donc plus réalistes. Bien entendu, cela implique aussi plus de calculs, et donc un coût plus important pour le processeur. Le CPU est bien plus actif après que nous ayons implémenté cette modification.

Question 8 : En amortissant la chute des particules, nous arrivons à stabiliser la simulation. L'amortissement simule l'interaction des particules avec l'air ambiant. C'est lui qui va réussir à compenser les interactions des particules entre elles et empêcher une oscillation de plus en plus importante comme observée précédemment.

Question 9 : Avec dt = 1/200, K = 300, et n = 20, nous arrivons à une simulation stable. Mais ces paramètres sont très coûteux en ressources car avec dt = 1 / 200 et n = 20, cela signifie que nous effectuons quatre mille fois la boucle « Update » par seconde. Plus tard, et sur des ordinateurs moins puissants, nous préférerons utiliser un dt de 1/50. Cela n'affecte pas trop la simulation mais permet de diminuer significativement le coût en ressources.

Question 10 : Le modèle de Mass-Spring-Damper est assez simple à utiliser dès lors que les formules à appliquer sont à disposition. En revanche, pour obtenir une simulation stable, le moindre écart dans les paramètres peut créer une simulation tout à fait exotique. En effet nous avons vu que diminuer la rigidité du mesh pouvait conduire à une simulation qui n'était pas réaliste, car le mesh se comportait plus comme une plaque pivotante que comme un rideau. Appliquer un amortissement trop fort rend la simulation trop longue pour être réaliste.

2 TP2

Question 1 : Baisser la valeur d'amortissement résulte en une accélération de la chute et en une baisse de la stabilité. En effet, l'amortissement sert à compenser les interactions entre les particules, qui font osciller le mesh, et à simuler une interaction avec l'air. Diminuer ces « frottements » accélère donc la chute du rideau, et les particules ont des interactions plus fortes qui déstabilisent la simulation. Après quelques recherches, la vitesse relative est simplement une soustraction des vecteurs vitesses du voisin par rapport à la particule.

Question 2 : Afin de simuler le sol, nous implémentons une limite de hauteur en dessous de laquelle les particules ne peuvent descendre, ayant pour effet de simuler un plan. Cette méthode nous a posé un petit problème lorsque les particules descendaient un peu plus bas que le seuil fixé, car nous supprimions alors toutes les forces sur les particules, et le manipulateur n'interagissait plus avec.

Question 3 : Afin de ne pas passer trop de temps sur une question qui ne nous semblait pas essentielle, nous avons simplement changé la couleur de notre manipulator, que nous montrons ci-dessous :

Question 4 : Le coefficient C, s'il est négatif, créé une attraction entre

le manipulator et le mesh. En revanche, s'il est positif, les deux objets se repoussent. Celui-ci peut permettre de simuler les propriétés adhérentes ou répulsives (à cause du contact) de différents outils qui pourraient être utilisés en chirurgie.

Question 5 :

Question 6 :

Question 7 :

Question 8 :

Question 9 :

Question 10 :

Question 11 : Afin d'atteindre les objectifs fixés, nous avons fixé une dizaine de points, un time step de 1/50e de seconde, et 20 boucles avant d'afficher le résultat.

3 TP3

Question 1 :

Question 2 : Afin de mieux repérer les particules fixées, nous avons décidé de les colorer en bleu.

Question 3 : Nous avons choisi de fixer des points à l'intérieur du foie, afin que la simulation soit plus réaliste. En effet, de cette manière nous préservons la « texture » du foie en autorisant la partie supérieure à bouger, comme si le foie était posé. Nous avons également souhaité conserver les lobes du fois afin de pouvoir les soulever.

Question 4 : Nous avons choisi de sauvegarder les particules fixées dans un fichier de type texte, nommé « save ». Si ce fichier existe déjà, il est chargé au lancement du programme. Celui-ci est écrasé à chaque appui sur la touche « s », ne laissant la possibilité que d'avoir une seule sauvegarde, à moins de la renommer manuellement.

Question 5 : Afin de permettre l'usage de la souris et du moniteur haptique au cours de la même simulation, nous avons ajouté une variable « mode », changée par l'appui sur la touche « m », qui permet d'autoriser le changement de position avec le moniteur haptique.

Question 6 : Afin d'avoir un contrôle plus précis, nous avons implémenté un facteur permettant d'amplifier ou de diminuer les mouvements du manipulateur. Celui-ci est contrôlé par « + » et « - » pour augmenter ou diminuer sa valeur, par seuil de 0.2.

Question 7 : Voici la manière dont nous avons choisi d'implémenter une forme d'inertie à notre manipulateur lorsque le bras articulé arrive en bout de course :

Question 8 : Nous avons choisi d'appuyer sur un des boutons du bras articulé afin de désactiver le mouvement du manipulateur. De cette manière nous pouvons remettre le bras à la position désirée avant de reprendre la manipulation.