TP Synthèse d'images: Simulation physique
- Animation de tissus -
CPE

durée - 4h

05 Janvier 2010

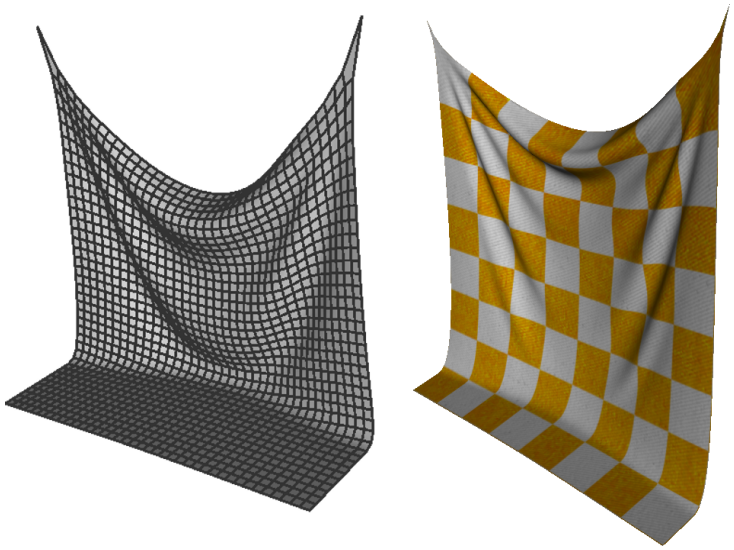

FIGURE 1 – Resultats possibles de la simulation de tissus.

1 But du TP

Le but de ce TP est d'implémenter le calcul et l'integration temporelle des forces de ressorts modélisant une simulation de tissus (voir fig. 1). L'integration se réalisera par la méthode de explicite d'Euler.

2 Prise en main de l'environnement

Le sujet reprend l'interface du TP de modélisation.

2.1 Classe Spring Mesh

Une nouvelle classe apparait : *spring mesh*. Il s'agit de la classe de gestion des ressorts liées.
Son fonctionnement est le suivant :
- *constructeur* : Construit une structure carree de N×N sommets
- *compute force* : Calcul les forces de gravitees et des ressorts et les stockent dans le conteneur *forces*. Cette methode est à compléter.
- *apply force* : Realise l'integration temporelle des forces ainsi que de la vitesse sur les positions. L'integration peut se réaliser suivant Euler explicite, et le pas de temps est passé en paramètre.

Cette classe permet egalement de gérer 3 raideurs de ressorts différentes.

2.2 Fonctionnement général

Dans le fichier *main* :

Un premier et unique passage dans la fonction *load model* est réalisé en début de programme (voir fig. 2).

```
//load the model
int load_model()
{
    try
    {
        smesh=spring_mesh(30);

        smesh.get_K_structural() = 1000;
        smesh.get_K_shear()      = 1000;
        smesh.get_K_bending()    = 1000;

        stepsize=0.015;


        opengl_drawer::generate_texture(1,"data/cloth2.ppm");
        glBindTexture(GL_TEXTURE_2D,1);


    }
    catch(std::string e)
    {
        std::cout<<"Exception: "<<e<<std::endl;
    }

    return 0;
}
```

construit maillage carré 30x30

initialise raideurs

pas de temps

chargement texture

FIGURE 2 – Fonction *load model* appelée une unique fois.

Ensuite la fonction *draw scene* est appelé en permanence au cours du déroulement .(voir fig. 3) C'est dans celle-ci que l'on réalise la MAJ des forces et l'integration temporelle.

```
//Draw the scene
int draw_scene()
{
    glEnable(GL_POLYGON_OFFSET_FILL);
    glPolygonOffset(5,5);


    glEnable(GL_LIGHTING);
    glColor3d(0.7,0.7,0.7);
    if(nav.is_texture())
        glEnable(GL_TEXTURE_2D);
    else
        glDisable(GL_TEXTURE_2D);


    //sauvegarde periodique du maillage a afficher
    //(independamment du calcul des forces et de l'integration)
    current_time = glutGet (GLUT_ELAPSED_TIME) / 1000.0;
    info_last_time += current_time - last_time;
    last_time = current_time;
    if (info_last_time >= 1.0 / FRAMERATE)
    {
        mesh_draw=smesh.to_mesh();
        mesh_normal=mesh_draw.normal_vertex();
        smesh.store_vertices();

        info_last_time=0.0;
    }

    //calcul des forces
    smesh.compute_force();
    //integration temporelle
    smesh.apply_force(stepsize);

    //affichage
    glDisable(GL_LIGHTING);
    glColor3d(0.2,0.2,0.2);
    if(nav.is_grid())
        cpe::opengl_drawer::draw(smesh); //affichage des quads en fil-de-fer

    glEnable(GL_LIGHTING);
    glColor3d(0.7,0.7,0.7);
    cpe::opengl_drawer::draw(mesh_draw,mesh_normal); //affichage du maillage

    return 0;
}
```

fonction appelée en permanence

MAJ des forces dans la classe spring_mesh

integration temporelle des forces
et de la vitesse

FIGURE 3 – Fonction *draw scene* appelée en continue.

3 Classe spring mesh

3.1 Intégration temporelle

L'intégration suivant la méthode d'Euler explicite suit l'algorithme suivant

```
for every vertices k
  speed[k] = (1-damping) speed[k] + dt forces[k]

for every vertices k
  vertices[k] += dt speed[k]
```

On notera que dans cette implémentation les vecteurs sont concaténés suivant x, y, z. Voici quelques exemples d'accès aux éléments :

```
forces[3*k+1];   // coordonnée y de la force du sommet k
speed[3*2+0];    // coordonnée x de la vitesse du second sommet
vertices[3*i+2]; // coordonnée z du sommet i
```

Question 1 *Complétez la méthode* apply force *afin de réaliser l'integration temporelle suivant la méthode d'Euler explicite.*

On notera que l'on peut forcer la valeur de la vitesse ou de coordonnées pour certains sommets dans cette méthode. Ainsi si l'on souhaite fixer entièrement le sommet k, il suffit de ne jamais mettre à jour ses coordonnées.

De même, en projettant la vitesse suivant un axe/plan donné, on pourra contraindre artificiellement le mouvement des sommets suivant ce degré de liberté.

Question 2 *Fixez deux sommets des coins du tissu.*

3.2 Mise en place des forces

La MAJ régulière des forces se réalise dans la méthode *compute force*. Pour l'instant seul une force de gravité est appliquée. Le tissu doit donc tomber avec une accélération constante.
Il convient de calculer les forces de rappels des ressorts et de mettre à jour le vecteur de forces pour chaque sommet. On désigne par s(ku,kv) le sommet s paramétré par (ku, kv).
On pourra séparer les ressorts suivant 3 types (voir fig. 4) :

- Ressorts structurels liant le sommet courant s(ku,kv) à
  - s(ku+1,kv)
  - s(ku,ky+1)
  - s(ku-1,kv)
  - s(ku,kv-1)
- Ressorts de cisaillement (shear) liant le sommet courant s(ku,kv) à
  - s(ku+1,kv+1)
  - s(ku-1,kv+1)
  - s(ku-1,kv-1)
  - s(ku+1,kv-1)
- Ressorts de courbure (bending) liant le sommet courant s(ku,kv) à
  - s(ku+2,kv)
  - s(ku,kv+2)
  - s(ku-2,kv)
  - s(ku,kv-2)

Ces 3 types de ressorts peuvent avoir des raideurs différentes en fonction du type de tissu simulé.

On notera que l'on pourra récupérer le vecteur de coordonnées du sommet s(ku,kv) par la syntaxe suivante :

```
unsigned int u=3*(ku+N*kv);
v3 x_current=v3(vertices[u+0],vertices[u+1],vertices[u+2]);
```

Question 3 *Complétez la méthode* compute force *de manière à calculer les forces des ressorts.*

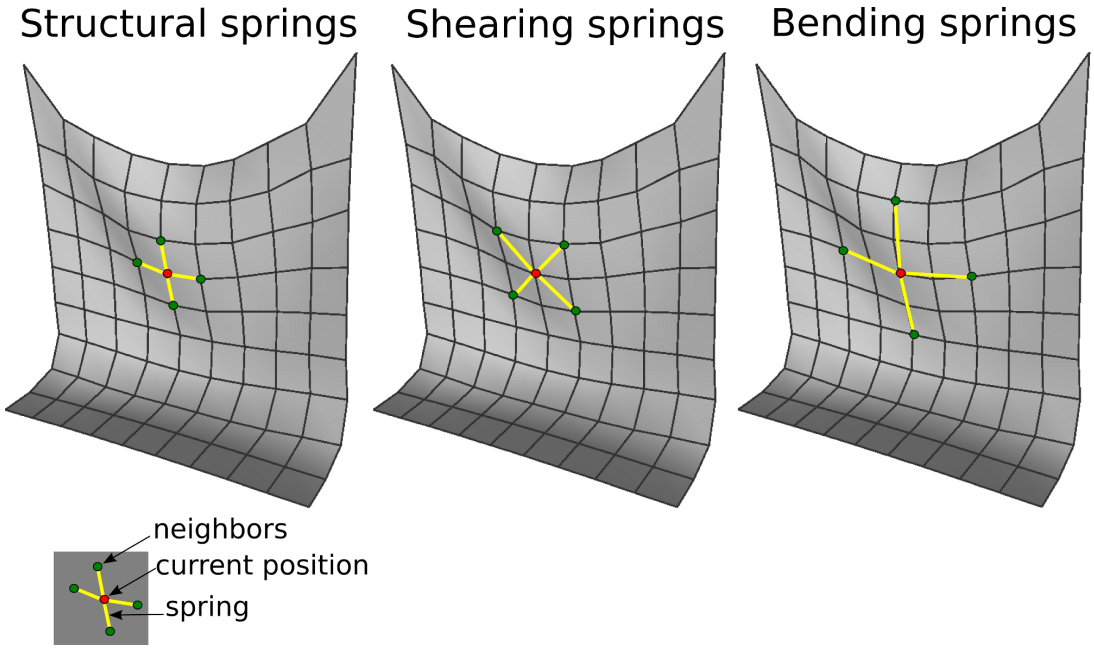

FIGURE 4 – Les trois types de ressorts appliqué au point courant (cercle rouge).

3.3 Etude complémentaire

Question 4 *Modélisez l'action d'un plan d'équation $z = -1$: Les sommets sont contraints à rester sur ce plan si $z < -1$ et les sommets en contacts pourront subire un frottement de type fluide.*

Question 5 *Observez l'influence de la subdivision du maillage sur la raideur. Augmentez la raideur et commentez son influence sur la stabilité du système. Que faut-il faire pour rendre le système plus stable ? Commentez.*

4 Implémentation sur GPU

Le second programme implémente une déformation calculée sur GPU. La position, vitesse et les normales des sommets sont passées en tant que texture. Trois shaders différents viennent remplir leurs valeurs.

Question 6 *Observez la structure du programme et différenciez la génération des textures liées aux positions, aux vitesses, aux normales et aux vraies coordonnées de textures.*

Quel est le type de shader utilisé ? Quelle est la géométrie 3D d'entrée envoyée à la carte graphique par le CPU ?

Question 7 *Complétez le shader de mise à jour de la vitesse en ajoutant les forces de rappels des ressorts. Comparez la vitesse d'execution par rapport à la version CPU précédente.*

La force exercée par le vent sur un tissu peut être modélisée par une force agissant dans la direction normale à la surface, et proportionnelle à l'angle entre la normale et la direction du vent. Pour cela, on peut considérer une force F_w telle que

$$F_w = K_w \ < \mathbf{n}, \mathbf{u}_w > \ \mathbf{n} \ ,$$

où K_w est une constante correspondant à l'intensité du vent, $\mathbf{n}$ est la normale à la surface, et $\mathbf{u}_w$ la direction du vent.

Question 8 *Ajoutez la force du vent dans votre simulation.*