Projet Maths pour l'info

—IMAC 1—

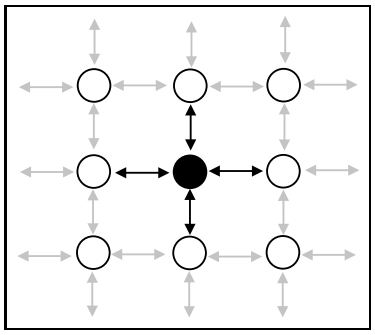
Système masse-ressort

Un système masse-ressort est un modèle décrivant le comportement d'un solide déformable soumis à des contraintes interieures et exterieures. Ce modèle permet de calculer le comportement de ces objets dans une scène dynamique, que ce soit pour du rendu réaliste ou des modélisations physiques. Ce projet traite ce sujet de façon légère sans trop approfondir les problèmes propres à cette méthode.

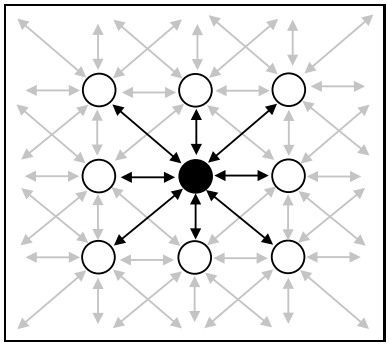
1 Position du problème

Le modèle masse-ressort décrit le comportement d'un solide déformable soumis à diverses contraintes dans une scène dynamique. Ce solide est décomposé en "particules" reliées par des interactions du type ressort linéaire. On attribue à chaque particule une masse et à chaque interaction un coefficient de raideur. On limite les interactions entre particules voisines. La méthode des systèmes masse-ressort consiste à mettre à jour la position des particules à chaque intervalle de temps dt. Pour celà, on calcule à chaque instant t et pour chaque particule la somme des forces qui lui sont appliquées. On en déduit par le principe fondamental de la dynamique son accélération, sa nouvelle vitesse et sa nouvelle position. Il s'agit donc d'une méthode itérative.

Dans le cadre de ce projet, nous traiterons le problème en 2d pour une question de simplicité d'affichage et de temps d'éxéctution. Nous définirons le voisinage d'un particule par son 8-voisinage (voir Figure *4-voisinage* et *8-voisinage*)



4-voisinage



8-voisinage

2 Modèle de ressort

Considérons un ressort reliant 2 masses m_i et m_j. Le ressort exerce sur les 2 masses une force $\overrightarrow{F_{ij}}$ du type :

$$\overrightarrow{F_{ij}} = c_{ij}(||\overrightarrow{x_{ij}}|| - L_{ij})\frac{\overrightarrow{x_{ij}}}{||\overrightarrow{x_{ij}}||} \tag{1}$$

où :

- c_{ij} est le coefficient de raideur du ressort reliant les masses i et j.
- $||\overrightarrow{x_{ij}}||$ est la distance entre les 2 masses à un instant t.
- L_{ij} est la distance entre les 2 masses au repos.

Il s'agit d'un modèle de ressort linéaire suffisament précis pour notre projet.

3 Principe fondamental de la dynamique

Le principe fondamental de la dynamique (2) nous permet de calculer l'accélération $\overrightarrow{a_i}$ de chaque masse m_i connaissant les forces qui lui sont appliquées.

$$m_i\overrightarrow{a_i} = \sum_j \overrightarrow{F_{ij}} \tag{2}$$

Il convient de répertorier les forces exterieures (le poids, les forces coulombiennes, etc ...) et les force interieures (les interactions entre masses, le vent ou les autres collisions). Pour notre projet, le principe fondamental de la dynamique peut s'écrire :

$$m_i\overrightarrow{a_i} = \sum_j c_{ij}(||\overrightarrow{x_{ij}}|| - L_{ij})\frac{\overrightarrow{x_{ij}}}{||\overrightarrow{x_{ij}}||} + \sum_k \overrightarrow{F_{ik}} \tag{3}$$

où les $\overrightarrow{F_{ik}}$ correspondent aux autres forces que celle appliquée par les ressorts. Parmi ces forces, nous pouvons inclure la force du poids $\overrightarrow{P_i} = -m_i\overrightarrow{G}$ (où $\overrightarrow{G}$ est la constante de gravitation universelle), nous pouvons éventuellement y inclure d'autres forces comme la force générée par du vent ou des forces ponctuelles.

4 Méthodes de résolution

Voici les structures de données conseillées :

```
struct particule{
  double masse;
  double positionX;
  double positionY;
  double vitesseX;
  double vitesseY;
  double accelerationX;
  double accelerationY;
  double forceX;
  double forceY;
  unsigned char couleur[3];
};

struct ressort{
  particule *particule1;
  particule *particule2;
  double distanceRepos;
  double tensionRessort;
};
```

Le programme principal doit contenir un tableau de ressorts et un tableau de particules.

La méthode de résolution se fait en 5 étapes (pour chaque itération):

1. calcul de la contribution des forces de chaque ressort sur les particules (initialisation des paramétres `forceX` et `forceY`). Cette étape se fait en parcourant le tableau des ressorts.
2. calcul des forces extérieures (le poids, le vent, ...) pour chaque particule (mise à jour des parametres `forceX` et `forceY`). Cette étape se fait en parcourant le tableau des particules.
3. calcul de l'accélération pour chaque particule.
4. mise à jour de la vitesse pour chaque particule.
5. mise à jour de la position de chaque particule avec détection de collisions.

5 Calcul des collisions

Il s'agit ici d'une introduction au calcul de collisions. Le problème étant suffisament complexe pour être lui-même un projet, nous utiliserons une méthode sommaire : si la nouvelle position calculée pour la masse m correspond à un support fixe, la masse m conserve son ancienne position et sa vitesse est réorientée (comme pour une réflexion ou un rebond). Il est conseillé d'appliquer cette collision en 2 étapes : une pour l'axe des x et l'autre pour l'axe des y.

Nous ne traitons que les collisions entre l'objet traité et les supports fixes, **nous ne traitons pas les collisions entre masses**.

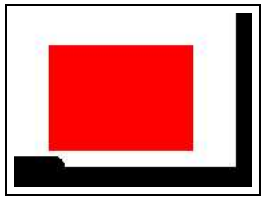
6 Mode opératoire

6.1 description de la scène

La scène utilisée et le système masse-ressort doivent être décrit dans une image au format *ppm* (voir paragraphe 8.1).

- Les pixels blancs correspondent à de l'air ou du vide.
- Les pixels noirs correspondent aux supports fixes indéformables.
- Les pixels rouges correspondent aux masses de l'objet.

Chaque pixel correspond à une masse. Le fichier *ppm* représente la masse au repos.



Exemple de scène

6.2 fichier de configuration

Les paramétres relatifs au système doivent être spécifiés dans un fichier de configuration (fichiers $.conf$) contenant dans l'ordre :

1. l'intervalle de temps `DT`.
2. La constante de gravitation universelle `GRAVITE`.
3. La masse des "particules" `MASSE`.
4. La tension des ressorts `TENSION_RESSORT`.
5. Le nombre d'itérations à effectuer `NB_ITERATIONS`.
6. Le nombre d'images à générer `NB_IMAGES`.

Exemple :

```
DT
0.005
GRAVITE
9.8
MASSE
0.1
TENSION_RESSORT
1500.0
NB_ITERATIONS
20000
NB_IMAGES
800
```

6.3 Rendu de la scène

Les images générées seront au format *ppm* (voir paragraphe 8.1). Elles seront numérotées dans l'ordre chronologique.

A noter : au repos, chaque masse occupe un pixel, en revanche, quand l'objet est déformé, plusieurs masses peuvent se retrouver dans le même pixel et par conséquent, certains pixels sont "vides" de masse. Ceci explique les traces blanches au milieu d'un objet.

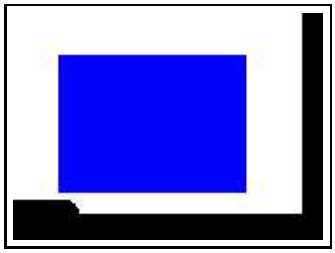


image 1

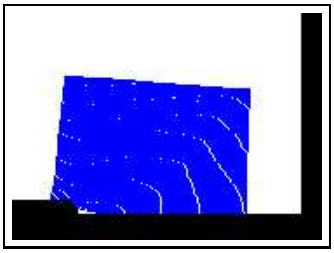


image 40

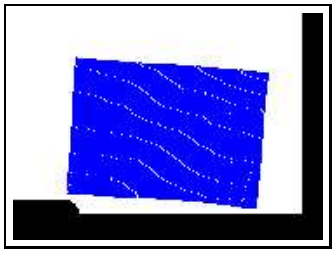


image 60

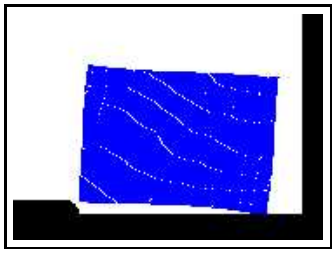


image 90

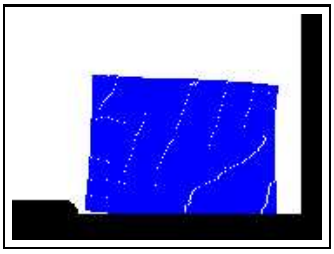


image 100

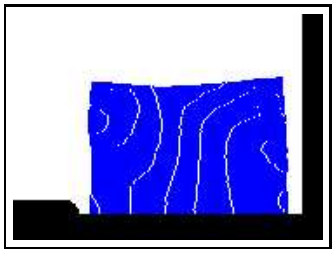


image 110

7 Travail demandé

Le but du projet est d'implémenter en langage `C` un programme permettant de lire un fichier de configuration et une scène entrée sur la ligne de commande et de générer une série d'images représentant la scènes à différents moments. Votre programme sera composé de fichiers `.c` et `.h`. Il se compilera avec un `makefile` et comportera une option `-h` pour l'aide. Toutes les options seront spécifiées sur la ligne de commande et nulle part ailleurs. Votre programme devra compiler et fonctionner sous Linux.

Vous devrez justifier le choix de vos méthodes dans un rapport d'eviron 5-10 pages. Dans ce rapport figureront également les observations et les difficultés rencontrées.

8 Documentation technique

8.1 fichier.*ppm*

Il s'agit d'un format d'images reconnu par divers éditeurs d'images comme *gimp* ou *ee* (electric eyes). Il en existe plusieurs variantes : couleurs/niveaux de gris, ASCII/binaire. Le format le mieux adapté à notre problème est le format couleur en binaire. Le fichier commencera donc avec l'en-tête correspondant.

Un pixel en couleur est défini par 3 nombres : un pour "la quantité" de rouge (R), un pour "la quantité" de vert (V) et un pour "la quantité" de bleu (B), noté RVB (RGB en anglais). Nous coderons ces quantités par des `unsigned char`. Un `unsigned char` est codé sur 8 bits, soit un octet. Ces nombres vont donc de 0 à 255.

Revenons au format *ppm*. Le premier nombre correspond au "numéro de variante" choisi, dans notre cas *P*6. Sur la ligne suivante, les deux nombres correspondent à la *largeur* et la *hauteur* de l'image. Enfin, le dernier nombre correspont à la valeur maximale que l'on puisse rencontrer (ici 255). Des commentaires peuvent être insérés dans l'en-tête, une ligne de commentaires commence par un #.
Chaque pixel sera écrit dans le fichier en procédant ligne par ligne, de gauche à doite, en commençant par le haut. Les pixels seront écrits sous forme de 3 octets écrits les uns à la suite des autres (RVB).

exemple:

```
P6
#commentaires eventuels
128 256
255
^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y
^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y
^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y
^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y
^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y
^@y^@y^@y ...
```

---

9 options

Il est conseillé de rajouter à votre programme quelques options. En voici quelques unes :

- Rajouter du vent aléatoire.
- Gérer les couleurs de l'objet pour un affichage optimal : on voit mieux les déformation sur un objet rayé rouge et vert.
- Traiter les collisions entre les masses.

Vous pouvez laisser libre cours à votre imagination, toute amélioration sera la bienvenue.

vous trouverez quelques informations complémentaires à l'adresse :

`http://www-igm.univ-mlv.fr/~vnozick/`

Bon courage.