

IFT1025 Automne 2009 — Devoir 2

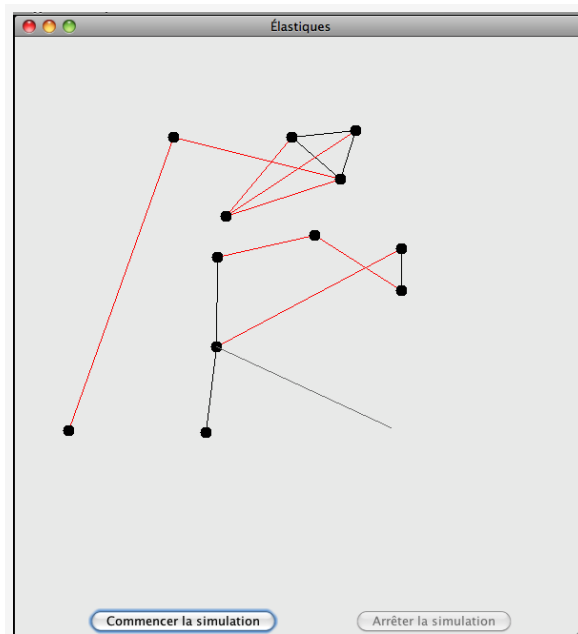
Miklós Csűrös

23 septembre 2009

À remettre avant 23 :59 mardi le 6 octobre par courriel à dift1025@iro.umontreal...

1 Les boules et les élastiques

Dans ce TP, vous avez à implanter une simulation pour modéliser le mouvement de boules liées par des élastiques. Le logiciel permettra de placer des boules et des élastiques sur un bord à l'aide d'un interface graphique, ainsi que de lancer et d'arrêter la simulation de mouvements des boules selon les lois physiques. Les classes de ce TP doivent être placées dans un package nommé `elastiques`.



GUI de l'application. Les élastiques tendus sont affichés en rouge, les autres en noir. L'utilisateur est en train de placer un élastique (la ligne en bas au droit finit au pointeur du souris).

1.1 Modélisation de boules et les élastiques

Une boule est représentée par un objet de la classe `Ball` qui est l'implémentation des interfaces `MovingObject` et `Drawable` (ces derniers vous sont fournis). L'interface `MovingObject` est pour représenter des objets avec position et vitesse en deux dimensions, voir les commentaires dans le code source. L'interface `Drawable` déclare seulement la méthode `void draw(Graphics2D g)`. Un élastique est représenté par un objet de la classe `RubberBand` qui implémente l'interface `Drawable`. Les attributs d'un tel objet sont la longueur neutre (ℓ ci-dessous), et deux objets de type `MovingObject` qui sont liés par l'élastique. Le comportement physique est modélisé par un objet de type `Board`. Cette classe implémente le calcul des forces sur chaque boule et celui de déplacements pendant une intervalle (courte) de temps δ . Cette classe vous êtes fournie.

1.2 Placement de boules et élastiques

L'affichage du bord est par la classe `BoardPanel` qui est une extension de `JPanel`. Le constructeur prend un `Board`. On peut placer une boule par le clic de souris sur une partie vide du bord. Pour placer un élastique, il faut sélectionner (par un clic) d'abord la première boule («boule gauche»), ensuite la deuxième boule («boule droite»). Les boules et les élastiques sont stockées dans deux `ArrayLists`. Dans l'implémentation, une classe interne `Cat` dans `BoardPanel` est utilisée pour travailler avec les événements du souris; cette classe est une sous-classe de `MouseAdapter`.

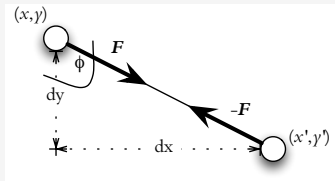
- ★ **Placement de boules.** Lors d'un clic, il faut vérifier si le clic se fait sur une boule (sélection) ou sur une partie vide (placement d'une nouvelle boule). Pour cela, utilisez les méthodes `closestObject` de `Board` et `contains` de `Ball`. Dans le cas du placement d'une nouvelle boule, la boule n'est pas sélectionnée.
- ★ **Placement d'élastiques.** Quand une boule est sélectionnée («boule gauche» d'un élastique), il faut afficher une ligne droite entre la boule sélectionnée et le pointeur du souris. Cette ligne est affichée (et suit le mouvement du souris) jusqu'à ce que l'utilisateur choisit la deuxième boule («boule droite») par un clic : elle peut être une boule déjà placée ou une nouvelle boule (clic sur une partie vide). Les deux boules définissent un nouvel élastique à placer sur le bord.
- ★ Le placement de boules et élastiques ne se fait pas quand la simulation est active (tester `isRunning()` du `Timer`, voir ci-dessous).

1.3 Simulation

La simulation est lancée ou arrêtée par les boutons en bas du JFrame de l'application. Seulement un des deux boutons est actif à la fois (utiliser `setEnabled(boolean b)` sur les boutons). Pour performer la simulation, on utilise un `Timer`. À chaque `ActionEvent` généré par le `Timer`, il faut recalculer les positions des boules et reafficher le bord (`repaint()`). Le `Timer` utilise une petite intervalle δ (10 millisecondes) : une action est générée à de telles intervalles, et les déplacements sont calculés avec le même δ (voir méthode `computeNewPositions(double d)` dans la classe `Board` : attention, l'argument `d` doit être en seconds).

1.4 Calcul de mouvement

Vous devez implanter le calcul de la force exercée par un élastique (méthode `getForce()` dans la classe `RubberBand`).



La vecteur de force $\mathbf{F}$ est déterminée par la longueur de l'élastique, c'est à dire, la distance entre les deux boules.

$$\|\mathbf{F}\| = \begin{cases} 0 & \text{si } d \leq \ell; \\ \alpha(d - \ell) & \text{si } d > \ell \end{cases},$$

où $d = \sqrt{(x - x')^2 + (y - y')^2}$ et (α, ℓ) caractérisent l'élastique : rigidité et longueur neutre. Pour calculer les coordonnées de la force, calculez d'abord l'angle ϕ (`Math.atan2` avec $dx = x' - x$ et $dy = y' - y$) : $\mathbf{F} = \|\mathbf{F}\| \cdot (\cos \phi, \sin \phi)$.

1.5 Application

L'application est lancée par la classe `Lancer` :

```
% java -cp Elastiques.jar elastiques.Lancer
```

2 Travail et soumission

Une grande partie (+600 lignes) du code source vous êtes fournie. Vous devez remplir la partie manquante, dénotée par des commentaires

```
// À IMPLÉMENTER
```

Notamment, vous devez travailler avec les classes suivantes.

- ▶ Classe `Ball` : méthodes des interfaces `Drawable` et `MovingObject`
- ▶ Classe `RubberBand` : toutes les méthodes indiquées
- ▶ Classe `BoardPanel` : traitement d'événements de souris et d'autres méthodes essentielles
- ▶ Classe `Lancer` : traitement d'événements de boutons pressés

Travail estimé : 200–250 lignes de code (incluant commentaires et lignes vides pour lisibilité).

Compilez un archive jar nommé `Elastiques.jar` qui contient les classes du package `elastiques`. L'archive doit inclure les sources `.java` et les classes compilées `.class`. Envoyez le fichier comme attachement dans un courriel à `dift1025@iro.umontreal....` Votre programme doit être fonctionnel, clair, commenté et original (plagiat=0).