



Collections en Smalltalk : Chenillard

Noury Bouraqadi

Préambule

Un chenillard désigne :

un mouvement lumineux qui se produit en allumant et éteignant successivement une série de lampes ou LED. L'effet se traduit par un déplacement de cette lumière dans un sens choisi, par exemple de gauche à droite. Comme l'illustre K 2000 (série télévisée), par effet Phi. [<http://fr.wikipedia.org/wiki/Chenillard>]
[...]

L'effet phi est la sensation visuelle de mouvement provoquée par l'apparition d'images perçues successives, susceptibles d'être raccordées par un déplacement ou une transformation. Le cerveau comble l'absence de transition avec celle qui lui semble la plus vraisemblable. C'est donc le résultat du traitement effectué par le système visuel. Cet effet est différent de la persistance rétinienne qui est un effet au niveau de la rétine. [http://fr.wikipedia.org/wiki/Effet_Phi]

1 Briques de base

En partant du code qui vous est fourni, vous complétez la classe `Guirlande` comme indiqué ci-dessous. Une guirlande est constituée d'une collection ordonnée de lampes. Chaque lampe est dotée d'une variable d'instance qui référence un afficheur graphique.

Dotez la classe `Guirlande` d'une seule variable d'instances nommée `lampes`. Puis, définissez les méthodes suivantes :

1. `lampes` retourne la collection de lampes.
2. `afficheurs` retourne une collection regroupant les afficheurs des lampes.
3. `demarrerAffichage` rend visible tous les afficheurs
4. `arreterAffichage` cache tous les afficheurs
5. `initialize`
 - initialise la variable d'instances `lampes` avec 10 lampes,
 - associe chaque lampe à un afficheur graphique
 - positionne les afficheurs sur une ligne espacée de 80 pixels
 - démarre l'affichage

Vérifiez que la guirlande se crée et s'affiche correctement. A ce stade, toutes les lampes sont éteintes.

2 Animation

Dotez la classe `Guirlande` des méthodes suivantes :

1. `eteindreToutesLesLampes`

2. `allumerSeulementLampesNumeros: desIndices` éteint toutes les lampes et allume celles dont les indices sont donnés en paramètre.
3. `unPasChenillardVersGauche` décale les états des lampes d'un pas vers la gauche. Chaque lampe récupère l'état (`estAllumee`) précédent de sa voisine de droite. Ainsi, l'état (`estAllumee`) d'une lampe à l'instant t_{n+1} est celui de sa voisine de droite à l'instant t_n . La lampe la plus droite récupère l'état précédent de la lampe à l'extrémité gauche de la guirlande.
4. `chenillardVersGauche`
 - allume les 3 lampes les plus à droite,
 - force le démarrage de l'affichage et
 - déclenche la répétition de `unPasChenillardDroiteGauche` pendant 50 fois, en utilisant la méthode fournie `executer: unBlock repetition: nombreDefois`

3 Afficheur à LED avec défilement

Dans cet exercice, vous allez développer un afficheur de texte avec défilement analogue à ceux qu'on peut en voir par exemple dans les bus. Dans un premier temps, définissez la classe `CaractereEnLED` qui affiche un caractère à l'aide d'une matrice de lampes organisées en 5 lignes par 3 colonnes. Vous pouvez exploiter pour cela la classe `Matrix` de Pharo. Vous devez afficher cette classe dans le *browser* afin d'identifier les méthodes qui pourraient vous servir. Mais voici quelques exemples pour vous aider.

- créer une matrice de 5 lignes et 3 colonnes remplie de nouvelles lampes :

```
|matriceDeLampes nombreLignes nombreColonnes |
nombreLignes := 5.
nombreColonnes := 3.
matriceDeLampes := Matrix
rows: nombreLignes
columns: nombreColonnes
tabulate: [ :indiceLigne :indiceColonne | Lampe new]
```
- parcourir la matrice en ayant accès aux indices des éléments :

```
matriceDeLampes withIndicesDo: [:lampe :indiceLigne :indiceColonne |
    Transcript
        cr;
        show: $();
        show: indiceLigne;
        show: $, ;
        show: indiceColonne;
        show: ') : '.
    lampe estAllumee
    ifTrue: [Transcript show: 'lampe allumee']
    ifFalse: [Transcript show: 'lampe eteinte']
]
```

- Pour obtenir un élément à une position donnée (avec `numeroLigne`, `numeroColonne` et `uneLampe` des variables)

```
numeroLigne := 3.
```

```
numeroColonne := 2.
```

```
uneLampe := matriceDeLampes at: numeroLigne at: numeroColonne
```

L’affichage d’une caractère est effectuée en allumant certaines lampes et en éteignant les autres. Afin d’améliorer la lisibilité veillez à minimiser l’espace entre les lampes et utilisez des afficheurs carrés. Pour cela, utilisez la méthode `afficherEnCarreDeCote: longueur` de la classe `AfficheurGraphiqueDeLampe` fournie.

Dans un second temps, définissez la classe `PanneauTexteEnLED` qui dispose d’une collection ordonnée d’instances de la classe `CaractereEnLED`. Outre l’affichage simple d’un texte, le panneau doit permettre le défilement de celui-ci, un peu à la manière de ce que vous avez fait avec la guirlande.