Animation basée sur Interpolation

proposé par Michael Emmi

Projet de Programmation 2012/2013 - 51IF2IK3 - 21 septembre 2012

1 Introduction

Le but du projet est d'implémenter un logiciel permettant de créer et de visualiser un dessin animé rudimentaire. Le programme sera composé d'une interface graphique pour rentrer les images décrivant l'animation, écrite en JFC/Swing. On engendrera des images complémentaires permettant d'avoir une animation fluide par interpolation.

Pour vous guider, nous vous proposons trois étapes d'avancement:

- implémenter une interface permettant de créer une image, de l'enregistrer, et de recharger une image déjà enregistrée,
- étendre l'interface pour créer, en modifiant l'image initiale, une séquance d'images décrivant l'animation; enregistrer cette séquence et la visualiser, et
- calculer les images complémentaires pour aboutir à un dessin animé que l'on visualisera.

On précise une décomposition possible des étapes ci-dessous.

2 L'Interface graphique pour la création des images

L'interface graphique doit être implémentée en utilisant la bibliothèque JFC/Swing.[1] Elle doit permettre de dessiner à la souris des formes simples (e.g. des carrés, des cercles). Le programme devra garder trace des figures rentrées par l'utilisateur.

2.1 Enregistrement et chargement d'une image

Étant donnée une image dessinée sur l'interface graphique, nous implémenterons une fonctionnalité qui nous permet de l'enregistrer dans un ficher, et une fonctionnalité pour regénérer un dessin d'un fichier enregistré. Lorsqu'il y a plusieurs façons possibles de le faire, nous vous proposerons le format standard SVG (scalable vector graphics, en anglais), basé sur des fichiers texte. Une documentation sur SVG est disponible à w3schools[2], mais puisque nous utiliserons seulement des figures simples, il su□t de suivre la syntaxe suivant:

[1] Un tutoriel de cette bibliothèque se trouve sur le site http://tecfa.unige.ch/guides/tie/pdf/files/java-swing.pdf.

[2] http://www.w3schools.com/svg/svg_reference.asp

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
    <rect id="0" x="100" y="200" width="50" height="50"
        stroke="black" fill="white"/>
    <rect id="1" x="340" y="13" width="20" height="55"
        stroke="black" fill="green"/>
    # ... etc.
</svg>
```

Chaque ficher `fichier.svg` doit commencer avec le ligne `<svg ...>` et doit finir avec le ligne `</svg>`. Au milieu, on liste des figures: ici on a mis quelques rectangles – chacun écrit comme `<rect ...>` – pour lesquels on a spécifié les coordonnées (e.g. `x="100"` et `y="200"`) et la taille (e.g. `width="50"`), accompagné des attributs supplémentaires (e.g. `id="0"`).

Ce format nous donne une façon facilement lisible et modifiable d'enregistrer les images, et de les embellir si vous en avez envie. Un autre avantage c'est que vous pouvez ouvrir et dessiner des fichiers SVG avec pas mal de logiciels (e.g. Firefox, Safari, Inkscape).

3 Extension aux séquences d'images

À la fin de la première partie, vous avez un programme permettant de dessiner une image à la souris et de l'enregistrer en format SVG sur le disque dur. Pour la deuxième partie, nous vous conseillons d'implémenter les fonctionnalités suivantes:

- étendre l'interface pour manipuler *plusieurs* images dessinées en même temps,
- étendre l'interface pour enregistrer et pour charger *plusieurs* images,
- ajouter la fonctionnalité permettant de visualiser une séquence d'images comme une animation

On élabore des étapes ci-dessous.

3.1 Enregistrement et chargement de plusieurs d'images

La aussi il y a encore plusieurs façons de le faire, nous vous proposons la façon suivante d'encoder des séquence d'images: on peut réutiliser le format SVG avec des tags de "groupe":

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
    <g id="image-1">
        <rect id="0" x="100" y="200" width="50" height="50" .../>
        <rect id="1" x="340" y="13" width="20" height="55" .../>
        # ... etc.
    </g>
    <g id="image-2">
        <rect id="0" x="105" y="200" .../>
        # ... etc.
    </g>
    # ... etc.
```

```
</svg>
```

C'est a dire que l'on va regrouper des images en utilisant des groupes, décrits par le tag `<g ...>`. Chaque groupe aura un attribut `id="image-X"` qui nomme une image, et chaque groupe contient l'image, décrite, comme avant, par des tags `<rect ...>` – pour des rectangles, par exemple. Notez que pour associer des figures entre plusieurs images/groupes, ça serait bien d'associer pour chaque figure un identificateur qui sera présent dans les différentes images; dans l'exemple ci-dessus, voyez le premier rectangle de chaque groupe, qui est identifié avec `id="0"`.

Donc, donné ce format, nous vous proposons d'étendre l'interface graphique pour enregistrer et charger des séquences d'images tout à coup.

3.2 Visualisation d'une séquence d'images

Étant donnée une séquence d'images ouvertes dans l'interface graphique, nous vous demandons de les animer, pour passer d'une image à l'autre, selon la séquence donnée. Puisque une animation raisonnable doit avoir environ 10–60 images montrées chaque seconde, il vous servira d'avoir une façon de générer des images intermédiaires automatiquement, au milieu chaque couple d'images-clé. La façon de générer des images intermédiaire est décrite dans la section suivante.

4 Interpolation des images-clés

Une façon standard de générer des images intermédiaires entre des images-clés est l'*interpolation* – voyez plus d'infos sur Wikipédia[3]. L'idée générale est de calculer les coordonnées intermédiaires des points spécifiques des figures de l'image. Notez que, pour des raisons de précision il vaut mieux faire les calculs avec le type `double` en Java.

Alors qu'il y a plusieurs façons d'interpoler, variées de l'effet sur l'animation générée, dans le cadre de notre projet il suffit q'une version simple. Étant données deux images-clés, avec un ensemble de figures en commun, on calcule une séquence des coordonnées, pour chaque figure, entre les coordonnées du figure dans les images-clés. Par exemple, supposons qu'on calcule 4 images entre deux images-clés contenant deux figures `A` et `B`, situées aux coordonnées `(1,1)` et `(6,6)`, respectivement, dans la première image, et `(6,6)` et `(1,6)` dans la deuxième, on calcule les coordonnées suivantes:

```
A:(2,2) A:(3,3) A:(4,4) A:(5,5)
B:(5,6) B:(4,6) B:(3,6) B:(2,6)
```

Quand les 6 images (les images générées avec les coordonnes calculées, encadrées par les deux images-clés) sont animées, l'une après l'autre, cela donne une animation fluide – en comparaison avec l'animation qui passe directement de la première image clé à la deuxième.

5 Documents à rendre

Tout au long du semestre vous aurez à rendre les documents suivants:

[3] http://fr.wikipedia.org/wiki/Interpolation_numérique

- spécification fonctionnelle/cahier des charges — semaine du 24 septembre 2012,
- spécification interne — semaine du 1er octobre,
- mode d'emploi — semaine du 10 décembre,
- javadoc — au fur et à mesure de la production de code.

Ces documents sont décrits dans la présentation générale de l'enseignement distribuée en début de semestre et disponible sur didel[4]. Après vous être connecté une première fois à la page de connexion[5] avec vos identifiant et mot de passe ENT et avoir été rattaché à votre projet, les diverses versions de votre code seront à gérer via le serveur svn[6] fourni par le script:

```
# initialization d'une copie locale
svn checkout --username *login* *url*

# mise à jour de la copie locale
svn update

# intégrer un ficher au dépôt
svn add *nom-du-ficher*

# publication des modifications locales
svn commit -m "*description des modifications*"
```

Les projets sont à rendre pour le lundi 17 décembre à 20h00. Les soutenances auront lieu pendant la session d'examen. Les détails de dates seront donnés sur le site didel de la matière.

[4] http://didel.script.univ-paris-diderot.fr/
[5] http://usvn.script.univ-paris-diderot.fr/login
[6] http://usvn.script.univ-paris-diderot.fr/svn/groupe/trunk