Images et sons

Table des matières

Applets, images et sons

Afficher une image dans une applet

URL

Du fait des limitations de sécurité, une applet ne peut pas accéder au disque dur de l'ordinateur client sur lequel elle s'exécute. Le fichier image qu'elle doit afficher doit être téléchargé à partir d'une URL.

Pour Java les URL sont représentées par la classe URL contenue dans le package java.net. On peut obtenir l'URL d'une applet en utilisant sa méthode **getCodeBase()**.

Téléchargement

La classe **Image** permet de représenter une image en mémoire. On peut créer une instance de cette classe en téléchargeant un fichier image au format gif ou jpg. Cette opération est réalisée par la méthode **getImage** qui peut prend deux formes :

- public Image getImage(URL url)
- public Image getImage(URL url, String name)

Dans la première forme on donne en paramètres une URL complète, dans la seconde on donne une URL de base et un chemin relatif sous forme de chaîne de caractères. Si le fichier image se trouve dans le même dossier que l'applet on créera un objet img de la classe Image en écrivant :

```
img=getImage(getCodeBase(),"monImage.jpg");
```

Affichage de l'image

L'affichage d'une image se fait comme tout affichage à partir d'un objet de la classe Graphics avec la méthode **drawImage**. Celle-ci peut prendre plusieurs formes parmi lesquelles on trouve :

- boolean drawImage(Image img, int x, int y, ImageObserver observer);
  L'image est affichée à la position de coordonnées (x,y) avec sa taille d'origine. L'objet observer doit implémenter l'interface ImageObserver, c'est le cas pour les descendants de la classe Component, donc pour l'applet ou un Canvas. On pourra donc désigner l'applet comme paramètre observer. Son rôle est de veiller à ce que le processus d'affichage se termine correctement.
- boolean drawImage(Image img, int x, int y, int width, int height, ImageObserver observer);
  L'image est affichée à la position de coordonnées (x,y) avec une largeur égale à width et une hauteur égale à height.

Exemple

Ecrivons une applet qui affiche une image titée d'un fichier dont le nom est passé en paramètre.

```
import java.applet.*;
import java.awt.*;
import java.net.*;

public class affimg extends Applet {

 Image img;

 public void init() {
  img=null;
  String S=getParameter("image");
  if (S!=null) {
   img=getImage(getCodeBase(),S);
  }
 }

 public void paint(Graphics g) {
  if (img!=null) g.drawImage(img,0,0,this);
 }

}
```

Jouer un son dans une applet

Téléchargement

La classe **AudioClip** permet de représenter des sons. On peut créer une instance de cette classe en téléchargeant un fichier son. Cette opération est réalisée par la méthode **getAudioClip** qui peut prend deux formes :

- public AudioClip getAudioClip(URL url)
- public AudioClip getAudioClip(URL url, String name)

Dans la première forme on donne en paramètres une URL complète, dans la seconde on donne une URL de base et un chemin relatif sous forme de chaîne de caractères. Si le fichier son se trouve dans le même dossier que l'applet on créera un objet son de la classe AudioClip en écrivant :

```
son=getAudioClip(getCodeBase(),"monSon.au");
```

Jouer un son

Pour faire entendre le son associé à un objet de type AudioClip on dipose des méthodes suivantes :

- **play** : pour jouer le son une fois
- **loop** : pour jouer le son en continu
- **stop** : pour stopper le son joué en continu

Exemple

Ecrivons une applet qui permet de jouer un son isolé et un son en continu grâce à deux boutons.

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class son extends Applet
 implements ActionListener {

 AudioClip ding;
 AudioClip fond;
 Button bding=new Button("DING");
 Button bfond=new Button("DEPART");
 boolean bruitFond=false;

 public void init() {
  ding=getAudioClip(getCodeBase(),"ding.au");
  fond=getAudioClip(getCodeBase(),"oiseau.au");
  add(bding);
  bding.addActionListener(this);
  add(bfond);
  bfond.addActionListener(this);
 }

 public void actionPerformed(ActionEvent e) {
  if (e.getSource()==bding) ding.play();
  else if (e.getSource()==bfond) {
   if (bruitFond) {
    fond.stop();
    bfond.setLabel("DEPART");
    bruitFond=false;
   }
   else {
    fond.loop();
    bfond.setLabel("STOP");
    bruitFond=true;
   }
  }
 }

 public void stop() {
  if (bruitFond) fond.stop();
 }
}
```

On a supposé que les fichiers ding.au et oiseau.au sont dans le même dossier que l'applet.

La méthode stop() de l'applet a été surchargée pour arrêter le son joué en continu lorsqu'on ferme la page html contenant l'applet.

Utiliser un MediaTracker

La classe MediaTracker

Le problème

Lorsqu'on construit un objet de la classe Image en utilisant la méthode getImage, le téléchargement du fichier image se fait en tâche de fond. L'objet obtenu n'est donc pas immédiatement opérationnel. La classe MediaTracker fournit les méthodes nécessaires pour savoir quand le téléchargement est effectivement terminé.

Utilisation d'un MediaTracker

Le constructeur de la classe MediaTracker est **MediaTracker(Component)**. On construit donc un objet MediaTracker trak en écrivant :

```
trak=new MediaTracker(this);
```

this représente le composant qui affiche l'image.

L'utilisation de cet objet se fait en deux temps :

- activation pour une image donnée avec un numéro d'identification, on utilise la méthode **addImage**.
- attente de la fin du téléchargement, on utilise la méthode **waitForId** dans une structure try...catch...

Ceci donne :

```
trak.addImage(img,0);
try {
 trak.waitForId(0);
} catch (InterruptedException e) {}
```

Image de fond pour une applet

Construisons une applet qui utilisera une image pour dessiner le fond. L'image sera affichée autant de fois que nécessaire pour remplir tout l'espace disponible.

Déclarations

Nous utiliserons 3 variables qui contiendont l'image, sa largeur et sa hauteur.

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class fond extends Applet {

 Image img;
```

```
int largeurImage;
int hauteurImage;
```

Téléchargement de l'image

Le nom du fichier image est passé en paramètres. Nous le récupérons dans la méthode init, puis nous créons l'objet img avec la méthode getImage et nous attendons la fin du téléchargement avec un MediaTracker. Nous lisons alors la largeur et la hauteur de l'image grâce aux méthodes **getWidth** et **getHeight** de la classe Image.

```
public void init() {
 String S=getParameter("fond");
 if (S!=null) {
  MediaTracker m=new MediaTracker(this);
  img=getImage(getCodeBase(),S);
  m.addImage(img,0);
  try {
   m.waitForID(0);
  } catch (InterruptedException e) {}
  largeurImage=img.getWidth(this);
  hauteurImage=img.getHeight(this);
 }
}
```

Dessin du fond

Le dessin se fait dans la méthode paint. Nous utiliserons une méthode auxiliaire nommée ligne pour afficher l'image autant de fois que nécessaire sur la largeur.

```
public void ligne(Graphics g, int y) {
 int x=0;
 while (x<getSize().width) {
  g.drawImage(img,x,y,this);
  x=x+largeurImage;
 }
}

public void paint(Graphics g) {
 if (img!=null) {
  int y=0;
  while (y<getSize().height) {
   ligne(g,y);
   y=y+hauteurImage;
  }
 }
}
```

Dessiner en mémoire

Il arrive parfois que l'on désire construire une image en mémoire avant de l'afficher. C'est le cas lorsqu'on veut créer des animations ou lorsqu'on veut conserver le contenu d'une zone de dessin sans structures particulières. Nous allons écrire une applet de dessin à la souris qui utilise cette possibilité.

Les déclarations

Nous utiliserons une variable img de type Image qui représentera l'image en mémoire et une variable gi de type Graphics qui sera le contexte graphique servant à dessiner dans l'image. Des variables oldx et oldy contiendront les coordonnées du dernier point dessiné ou -1 si celui-ci n'existe pas. Ceci nous donne :

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class dessin extends Applet
 implements MouseListener, MouseMotionListener {

  Image img=null;
  Graphics gi=null;

  int oldx=-1;
  int oldy=-1;
```

Les interfaces MouseListener et MouseMotionListener nous permettront de gérer les évènements souris.

Méthode init

La méthode init de l'applet nous permettra de construire img et gi en utilisant la méthode createImage de l'applet et la méthode getGraphics de l'image obtenue. On obtient :

```
public void init() {
 addMouseListener(this);
 addMouseMotionListener(this);
 img=createImage(getSize().width, getSize().height);
 gi=img.getGraphics();
}
```

Méthodes paint et update

La méthode paint se contente de dessiner l'image. Nous réécrivons la méthode update pour éviter les effacements d'écran à chaque appel de repaint.

```
public void paint (Graphics g) {
 g.drawImage(img,0,0,getSize().width,getSize().height,this);
}

public void update (Graphics g) {
 paint(g);
}
```

Gestion de la souris

Nous devons enfin écrire les méthodes exidées par les interfaces de gestion de la souris. Nous effectuerons les opérations suivantes :

- lors de l'appui sur un bouton, initialiser oldx et oldy et dessiner le point correspondant dans l'image

- lors des déplacements avec le bouton appuyé, relier le point montré par la souris à celui correspondant à oldx et oldy, puis mettre ces deux variables à jour
- lors du relachement de la souris, relier le point montré par la souris à celui correspondant à oldx et oldy, puis mettre ces deux variables à -1.

On obtient le code suivant :

```
            // Gestion de l'événement "bouton de la souris enfoncé"
            public void mousePressed(MouseEvent evt) {
             oldx = evt.getX();
             oldy = evt.getY();
             gi.drawLine(oldx,oldy,oldx,oldy);
             repaint();
            }

            // Gestion de l'événement "bouton de la souris relevé"
            public void mouseReleased(MouseEvent evt) {
             if (oldx>=0 && oldy>=0) {
              gi.drawLine(oldx,oldy,evt.getX(),evt.getY());
              oldx=-1;
              oldy=-1;
              repaint();
             }
            }

            public void mouseEntered(MouseEvent e) {}

            public void mouseExited(MouseEvent e) {}

            public void mouseClicked(MouseEvent e) {}

            // Gestion de l'événement "déplacement souris avec bouton
enfoncé"
            public void mouseDragged(MouseEvent evt) {
             if (oldx>=0 && oldy>=0) {
              int newx=evt.getX();
              int newy=evt.getY();
              gi.drawLine(oldx,oldy,newx,newy);
              oldx=newx;
              oldy=newy;
              repaint();
             }
            }

            public void mouseMoved(MouseEvent evt) {}
```

Exercices

1. Visionneuse d'images
   Ecrire l'applet Visio qui affiche des images (nom de fichiers passés en paramètres) sélectionnées dans une liste.

2. Machine à sous
   Ecrire l'applet slot qui simule une machine à sous.

3. Dessin en couleur
   Ecrire l'applet dessin en s'inspirant de l'applet étudiée dans la partie "Dessiner en mémoire."