Drag & Drop avec Style et l'API Swing

par Romain Guy

Date de publication :

Dernière mise à jour :

Drag n' Ghost : Démonstration de Drag & Drop avec l'API Swing.

I - Démonstration avec Java Web Start

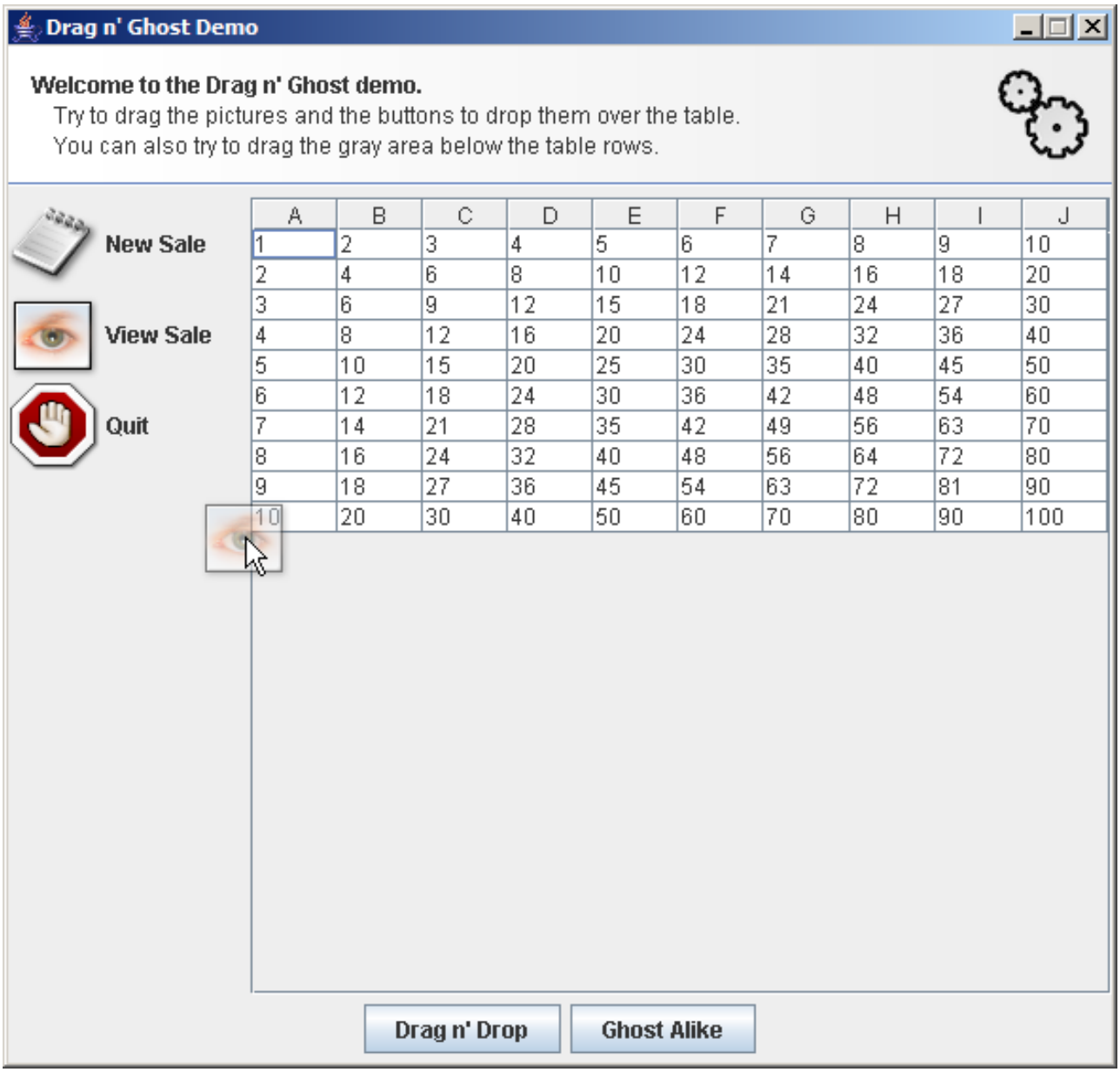
Prérequis *: Pour essayer cette démonstration vous devez disposer de Java Web Start (compris avec le JRE et le JDK).*

Lancez la démonstration

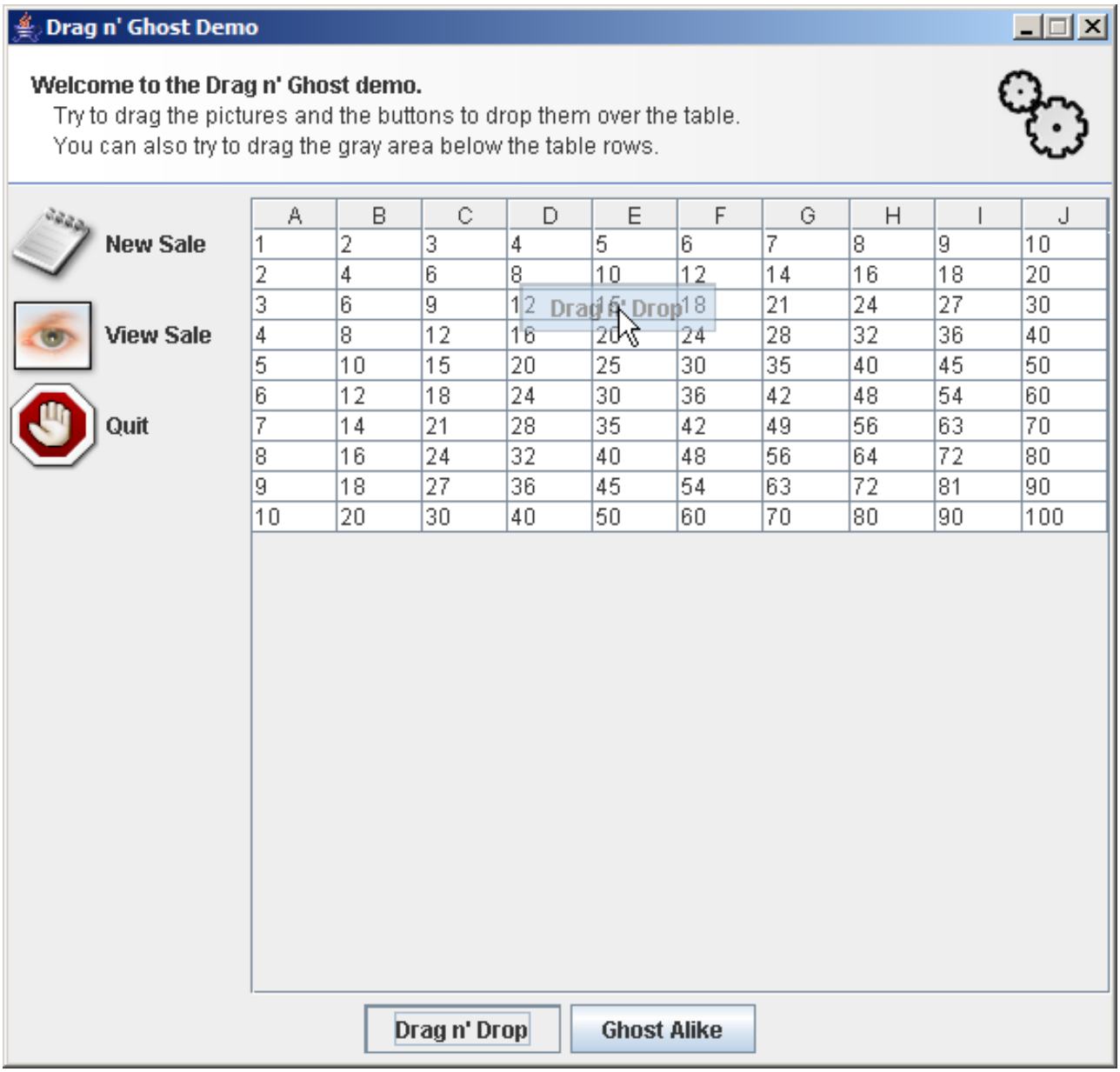
II - Explications

Une manière très sympa de créer des interfaces graphiques *user friendly* est de permettre de faire du drag & drop de tout ce que l'on veux dans, depuis et à l'intérieur même de l'application. MacOS X est l'exemple parfait d'une bonne utilisation de drag & drop. A chaque fois que j'ai essayé de *dragger* quelque chose et de le *dropper* dans quelque chose d'autre, ca a marché. Sauf biensûr si vous essayez de faire quelque chose de stupide du style *dropper* un fichier texte sur l'icône d'un jeu. Biensûr Windows et Linux permettent aux applications d'implémenter du drag & drop, mais il leur manque quelque chose que MacOS X offre déjà : des effets visuels vraiment cools. Par exemple, *dragger* une image depuis Safari, le navigateur web, vous laisse voir une jolie vignette translucide de l'objet.

Comment pouvons nous nous débarasser des curseurs de drag & drop simplistes offerts par Java ? La solution réside dans la démo Drag n' Ghost que je vais vous montrer. Voyez par vous même, vous n'aimez pas cette jolie image translucide ?

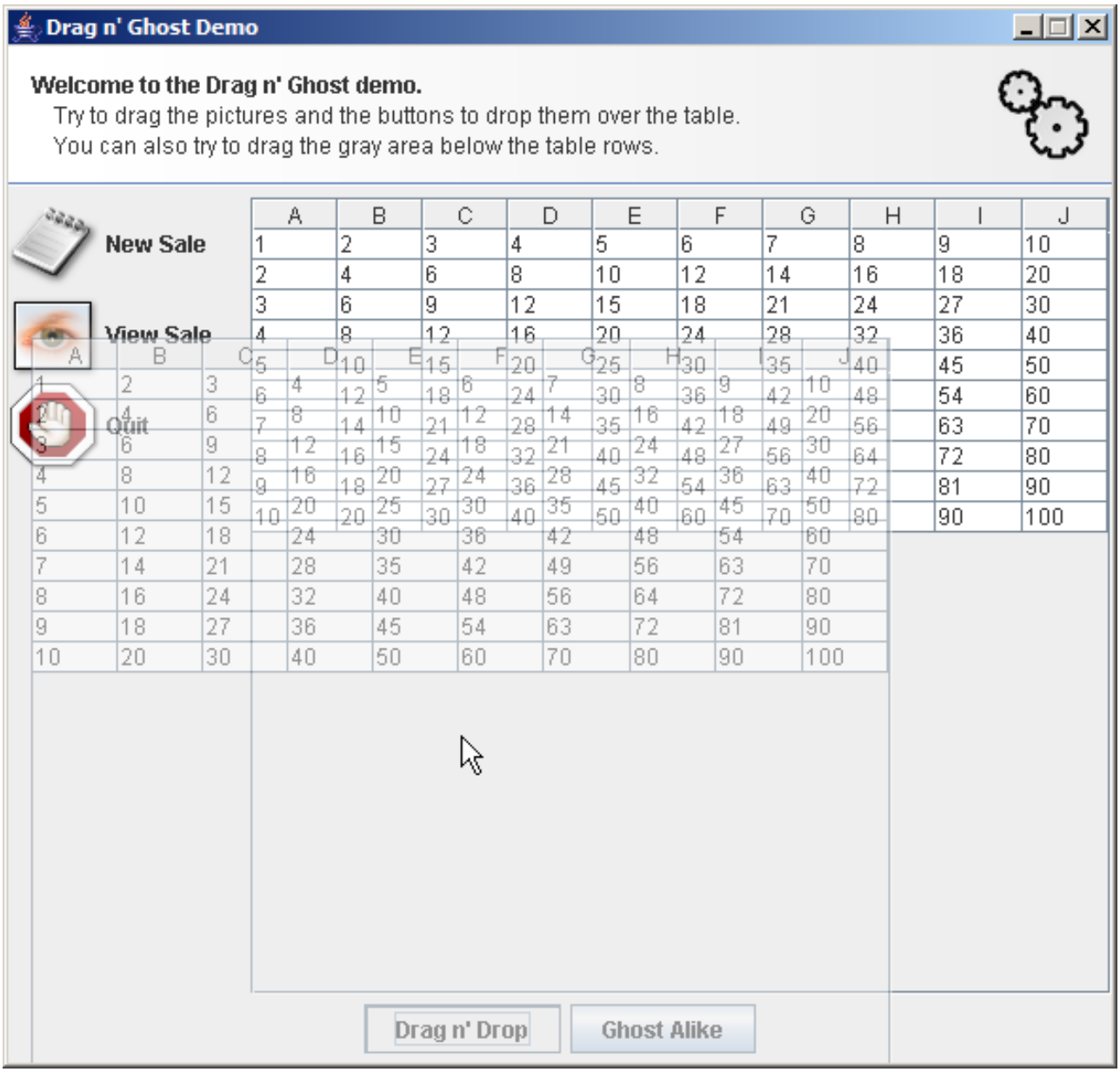


Dragger une image c'est bien, mais ce ne serait pas mieux de pouvoir *dragger* n'importe quel composant tout en gardant ce bel effet ? Aucun problème :



A	B	C	D	E	F	G	H	I	J
1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Toujours pas convaincus ? Voyons à quoi ca ressemble de *dragger* toute une JTable :



Maintenant que j'ai votre attention, voyons voir comment utiliser ca dans vos applications. L'astuce est d'utiliser un *glass pane* pour dessiner l'image translucide es composants par dessus l'UI. Vous devez donc mettre un *GhostGlassPane* à la fenêtre et attacher un *GhostDropAdapter* et un *GhostMotionAdapter* au composant que vous voulez *dragger* :

```
GhostGlassPane glassPane = new GhostGlassPane();
setGlassPane(glassPane);

JLabel label = new JLabel("Component adapted");
label.addMouseListener(new GhostComponentAdapter(glassPane, "action_1"));
label.addMouseMotionListener(new GhostMotionAdapter(glassPane));

JButton button = new JButton("Picture adapted");
label.addMouseListener(new GhostPictureAdapter(glassPane, "action_2", "image.png"));
label.addMouseMotionListener(new GhostMotionAdapter(glassPane));
```

Et voilà, c'est aussi simple que ca. Vous pouvez noter que nous n'utilisons pas *GhostDropAdapter* mais ses deux sous-classes *GhostComponentAdapter* et *GhostPictureAdapter*. La première crée l'image fantôme en dessinant le composant source dans une image offscreen alors que la seconde utilise l'image fantôme que vous lui fournissez.

Chaque *GhostDropAdapter* peut enregistrer des *GhostDropListeners* qui sont invoqués lors du drop. L'évènement est une instance de *GhostDropEvent* qui vous donne le nom de l'action et la localisation du drop. Tant que cette position est dans les coordonnées de l'écran, il est conseillé d'utiliser *AbstractGhostDropManager* qui propose deux méthodes intéressantes.

Un tel manager est attaché à un composant (voir le constructeur) appelé la cible. Deux méthodes vous permettent de gérer le composant : *isInTarget()* vérifie si la position du drop est à l'intérieur des bornes du composant cible et *getTranslatedPoint()* traduit les coordonnées écran dans le système de coordonnées de la cible. *GhostDropManagerDemo* vous montre comment créer un manager personnalisé. Dans cet exemple, le manager attends un drop sur la JTable et montre le nom de l'action :

Cependant, l'ensemble des composants GhostDrop souffrent de deux problèmes. Tout d'abord, les performances peuvent être très mauvaises quand vous *draggez* un ghost sur une grande fanêtre. Je ne l'ai vu se produire que sur mon ordinateur, mais je ne doute pas que celà peut se produire aussi sur les votres. Je ne vois pas comment optimiser le rendu à part peut-être utiliser une VolatileImage (et je ne suis pas certain du résultat). Le second problème se produit quand le focus est donné à une autre fenêtre alors que vous êtes en train de *dragger* quelque chose. Je pense que celà peut être résolu très facilement en écoutant le focus de la fenêtre parent.

III - Téléchargements

Version PDF (mirroir HTTP)

Version HTML/ZIP (mirroir HTTP)

Code source (mirroir HTTP)