Transfert de Données: drag and drop et {couper, copier} - coller

Sylvain Malacria - www.malacria.fr
IHM Master 1 informatique - Université de Lille 1

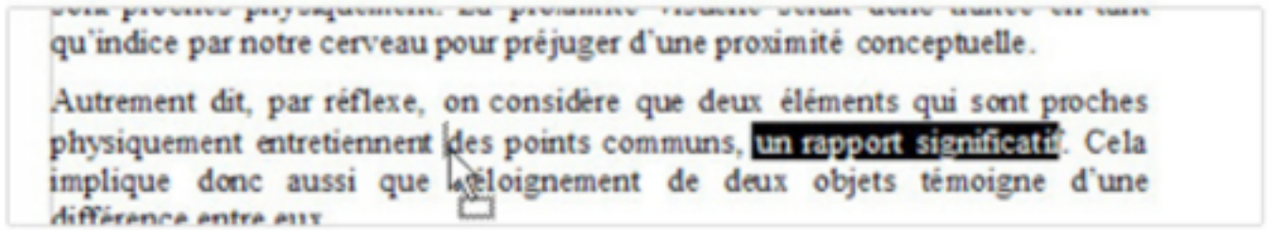
Adapté de Géry Casiez

Drag and drop

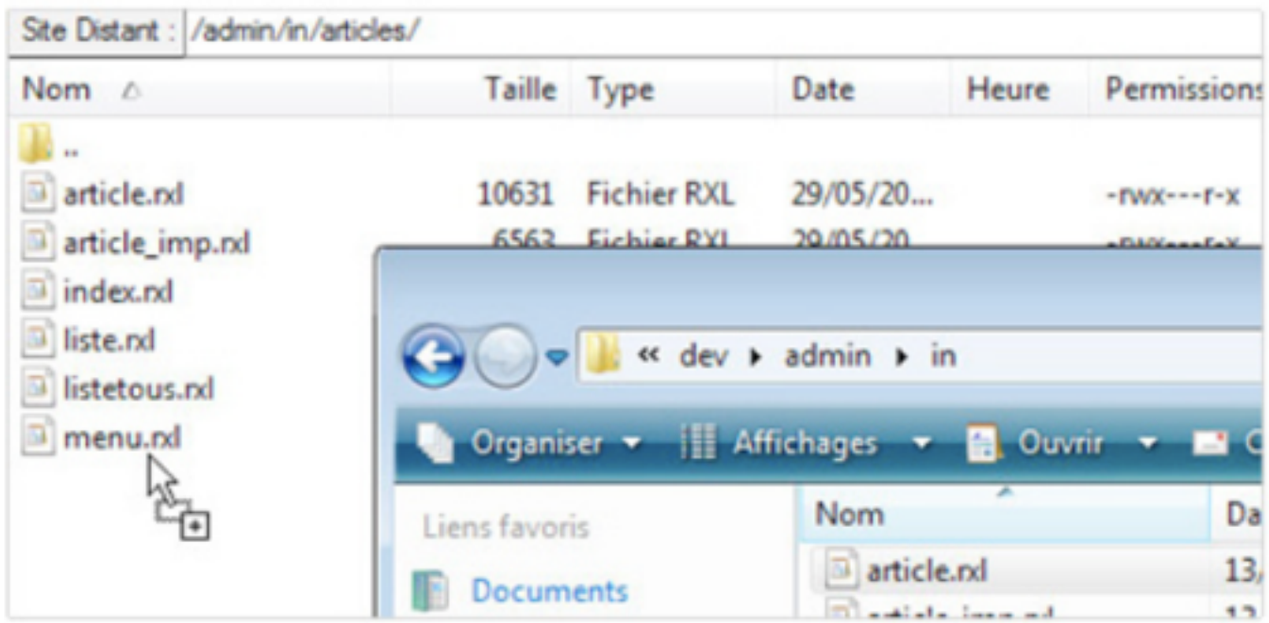
Drag and drop (DnD): glisser-déposer ou cliquer-glisser en français est une technique d'interaction qui permet de prendre un objet et de le déplacer à un autre endroit ou sur un autre objet.

Le Drag and drop et le couper/copier-coller (CCP) sont des fonctionnalités essentielles de la plupart des applications.

Exemples


qu'indice par notre cerveau pour préjuger d'une proximité conceptuelle.

Autrement dit, par réflexe, on considère que deux éléments qui sont proches physiquement entretiennent des points communs, un rapport significatif. Cela implique donc aussi que l'éloignement de deux objets témoigne d'une



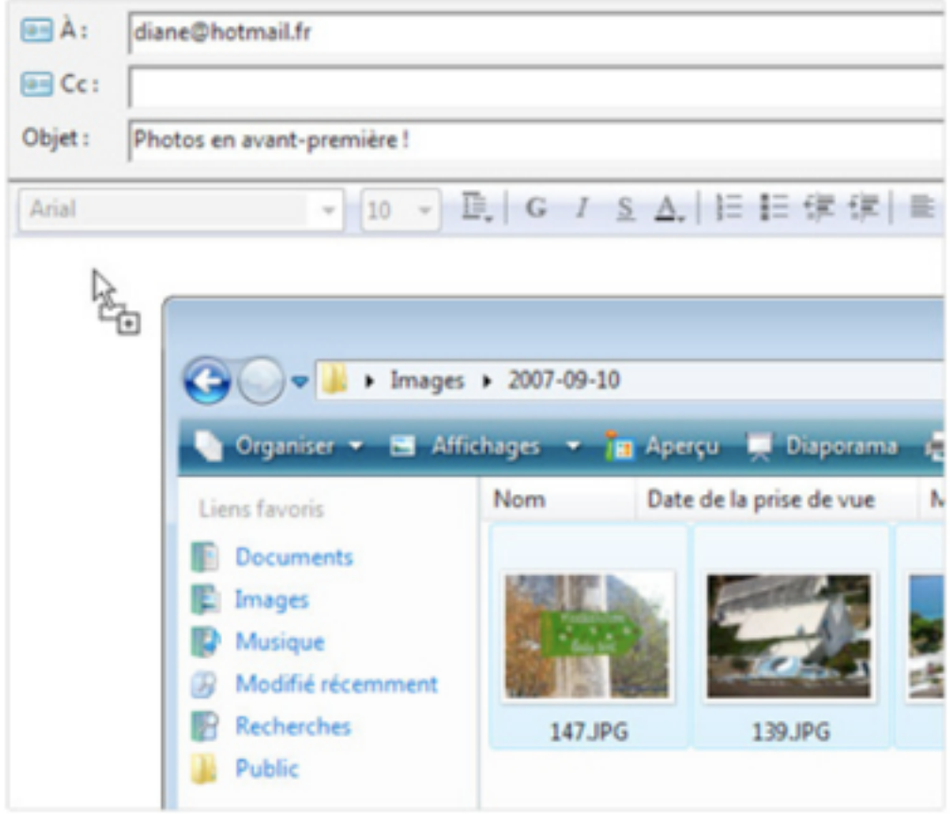
Exemples

- Déplacer une icône à la corbeille pour supprimer un fichier, sur imprimante pour imprimer…
- Déplacer un fichier d'un répertoire à un autre ou d'une fenêtre à une autre
- Réorganisation de palettes d'outils dans un logiciel de dessin
- Déplacer un fichier sur une fenêtre pour voir son contenu
- Déplacer des objets (dessin vectoriel)
- Déplacer / copier du texte d'un document à un autre
- Ajouter des objets à une liste
- Déplacer une commande sur un objet
  (Exemple: spécifier la couleur d'un objet)
- Créer un lien d'une portion de document à une autre

Réel outil d’optimisation des interactions

Attachement de fichiers dans un e-mail

Entrée par l’action vs entrée par l’objet



Manipulation directe

3 principes fondamentaux:

- Représentation permanente des objets d'intérêt
- Actions rapides, incrémentales et réversibles, dont les actions sur les objets sont immédiatement visibles
- Utilisation d'actions « physiques » (ex: pointer, déplacer) plutôt que de commandes à la syntaxe complexe

Exemple: déplacer un fichier sur une corbeille plutôt que de taper la commande rm

Traitements de texte, tableurs, jeux vidéos, CAO

Couplage perception/action

Agir pour percevoir

- Perception de la profondeur par des mouvements de la tête
- Perception de la texture d'un objet en déplaçant le doigt sur sa surface

Percevoir pour agir

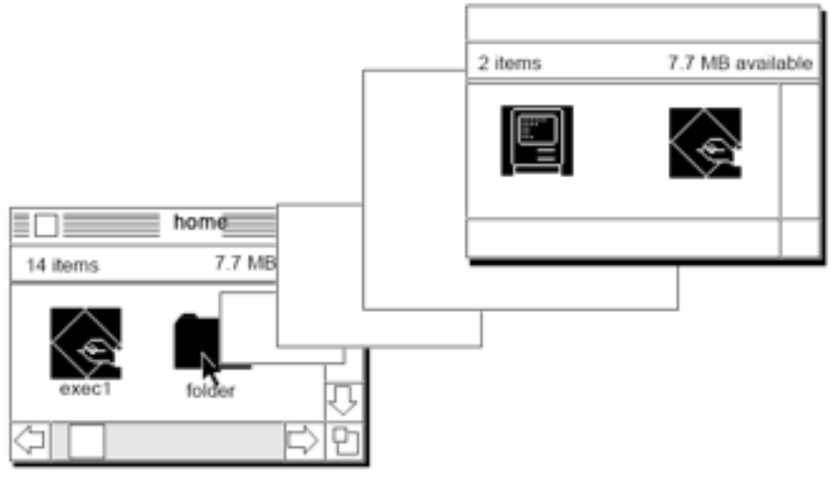
- Ajuster les mouvements du bras pour saisir un objet

Importance de la continuité de ce couplage

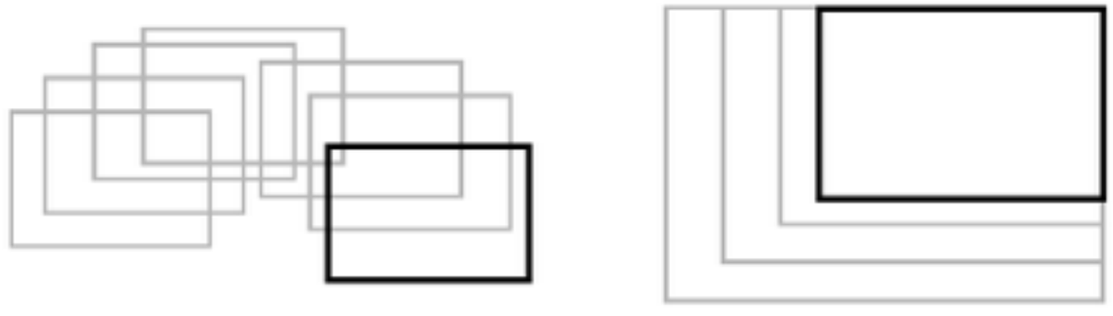
Le feedback (retour d'information)

Pointage

Sélection



Tracé, déplacement,
transformation



Manipulation directe

Sketchpad 1962

Manipulation directe

Avantages

+ Les novices peuvent apprendre rapidement, habituellement par la démonstration d'un utilisateur expérimenté
+ Les experts peuvent travailler plus rapidement en menant une large gamme de tâches
+ Les utilisateurs peuvent voir immédiatement le résultat de leurs actions
+ Les utilisateurs prennent confiance parce que l'interface est compréhensible et parce que les actions peuvent être facilement inversées
+ Les utilisateurs gagnent en confiance et maîtrise parce qu'ils sont les initiateurs de leurs actions et qu'ils peuvent anticiper les réponses de l'interface
+ Les messages d'erreur sont rarement nécessaires

Manipulation directe

Inconvénients

- Utilise beaucoup de place par rapport à l'interface en ligne de commande
- Apprentissage de la signification des icônes: autant d'apprentissage sinon plus que l'apprentissage d'un mot.
- Les représentations des icônes peuvent être trompeuses ou l'utilisateur peut ne pas comprendre la métaphore visuelle (utilisation de concepts connus par l'utilisateur)
- Demande parfois plus de temps de prendre la souris que d'utiliser le clavier

Historique

Inventé par Xerox PARC

Popularisé par Apple en 1984

Maintenant utilisé par toutes les interfaces graphiques

Java Swing

- Apparu dans le JDK 1.2 (~ 10 classes du package java.awt.dnd.*) DragGestureRecognizer, DragGestureListener, DragSourceContext
- Nouvelle API plus simple et plus légère dans le JDK 1.4 TransferHandler, Transferable
- Nouvelles fonctionnalités dans JDK 1.6

Apparition dans le WEB 2.0

Types de transferts

On parle plus généralement de transferts de données

- Utilisation de la manipulation directe
- Utilisation du presse-papier par l'utilisation de touches ou icônes de raccourcis

Déplacer (move) (clic souris enfoncé)

- Action par défaut
- Couper-coller au clavier (crtl-x, crtl-v)

Copier (copy) (clic souris enfoncé + ctrl)

- Appui de la touche ctrl après avoir initié le drag
- Copier-coller au clavier (crtl-c, crtl-v)

Lien (link-alias) (clic souris enfoncé + crtl + shift)

Type de données transférables

N’importe quel type:

- Texte
- Images
- Liste de fichiers
- Vidéos
- Son
- Couleurs
- Personnes…

Différents contextes

Transférer des informations d'une application en interne

- Entre composants

Transférer des informations en externe (entrer ou sortir des informations de ou vers l'application)

- Entre applications Java
- Entre applications natives et applications Java

3 acteurs

3 principaux acteurs

- Une source (composant d'où l'objet est déplacé)
- Le système (toolkit et/ou système de fenêtrage sous-jacent)
  – prise en charge des données
- Une destination (composant sur lequel l'objet est déplacé)

Etapes

Le début de l’interaction (activé en général par un glissement de souris, un bouton particulier étant maintenu enfoncé) durant lequel l’objet d’un drag particulier est spécifié, alors que tous les mécanismes nécessaires à sa réalisation sont mis en place

Le glisser (drag) qui consiste en un déplacement de souris, bouton toujours enfoncé, vers des cibles potentielles en les notifiant pour qu’elles puissent se déclarer intéressées

Le déposé (drop) qui consiste à effectivement acheminer les données de la source à la cible

La finalisation qui permet aux sources et aux cibles de redevenir passives pour le DnD

Exemple d'utilisation

Copie de texte d'une liste à un champ de texte

1) sélection d'un item de la liste (source)

2) en cliquant sur le texte, déplacement de la souris: début du geste de drag (drag gesture)

3) la liste regroupe les informations à exporter et spécifie le type d'action supportée: copie, déplacement, lien (source action)

4) Lors du déplacement des données, Swing calcule en permanence l'emplacement et met à jour l'affichage

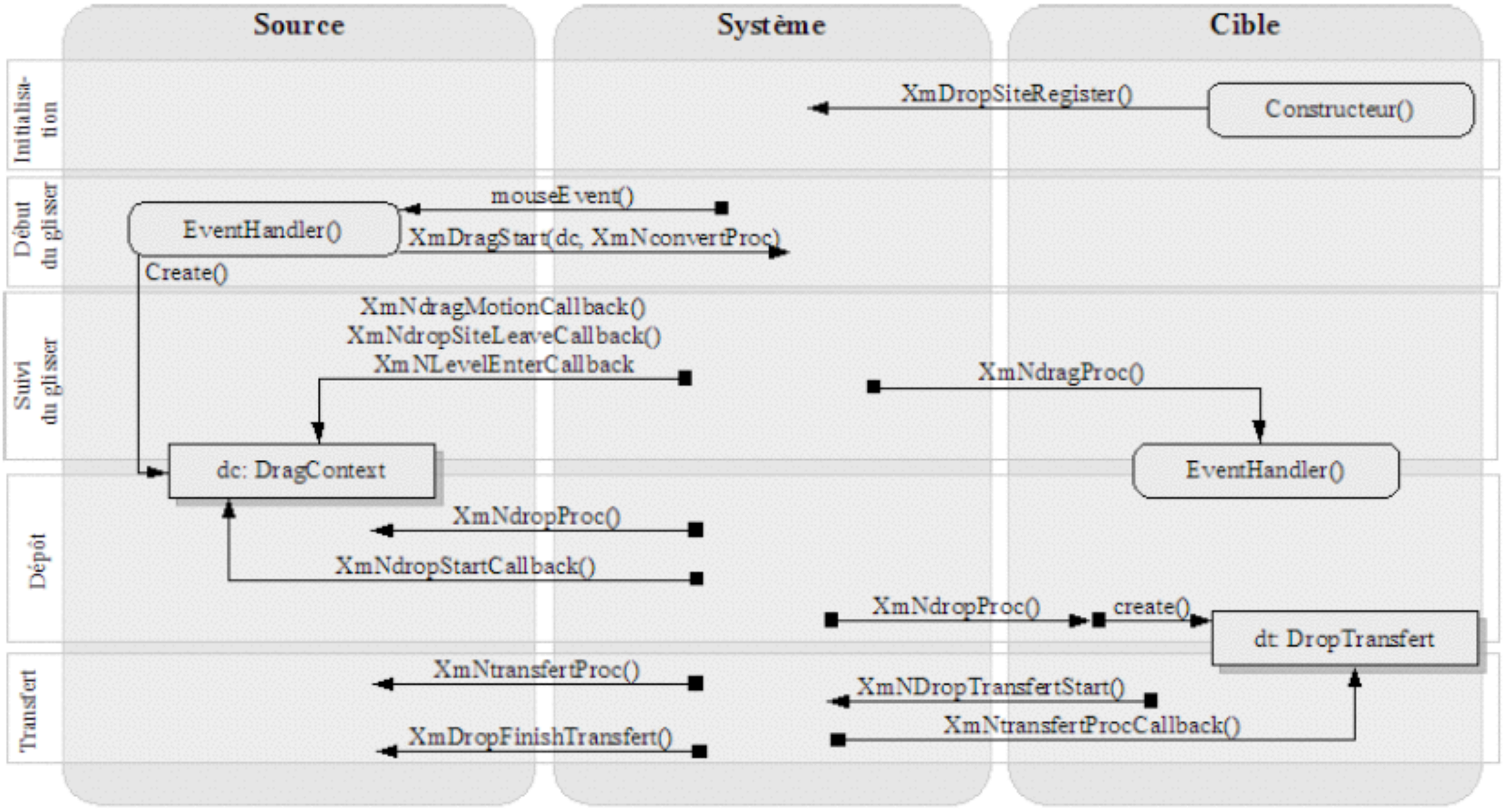
5) Si l'utilisateur appuie sur shift ou ctrl, cette action de l'utilisateur (user action) est aussi prise en compte

6) Quand l'utilisateur arrive sur le champ de texte, la cible (target) est interrogée en permanence pour voir si elle accepte ou refuse un drop potentiel. La cible fournit un retour d'information pour montrer l'emplacement de drop (drop location)

7) Quand l'utilisateur relâche le bouton, le composant de texte inspecte les actions déclarées dans la source et les actions de l'utilisateur

8) Le champ de texte importe les données

Fonctionnement du DnD sous X-Window



Le DnD dans le JDK

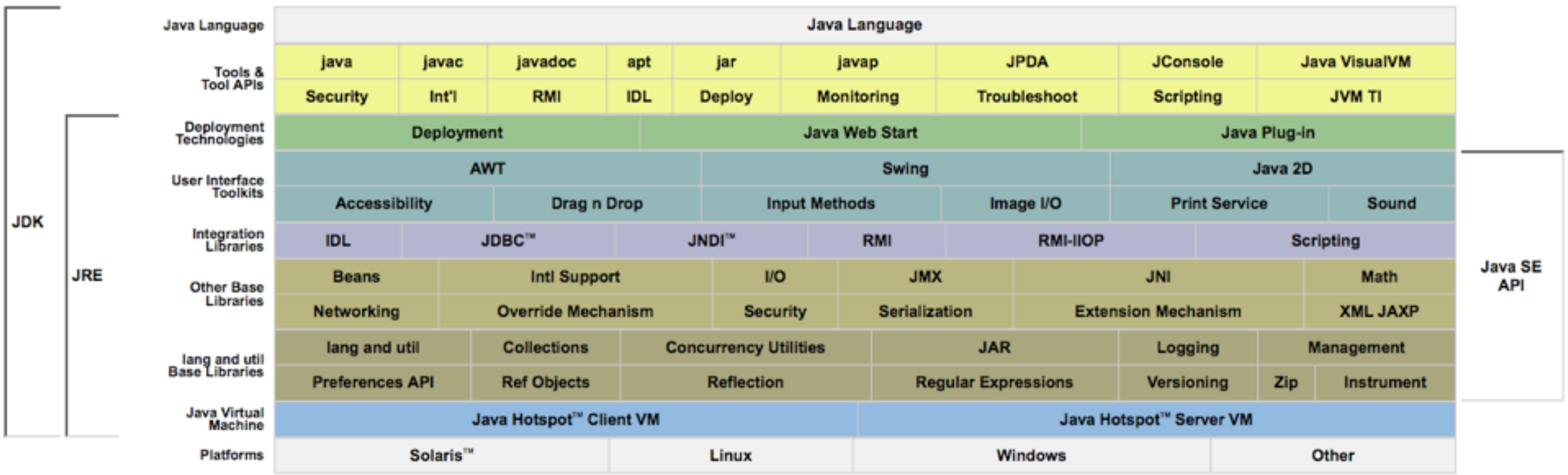
Se situe dans AWT et Swing

Packages java.awt.dnd et java.awt.datatransfer

javax.swing.TransferHandler

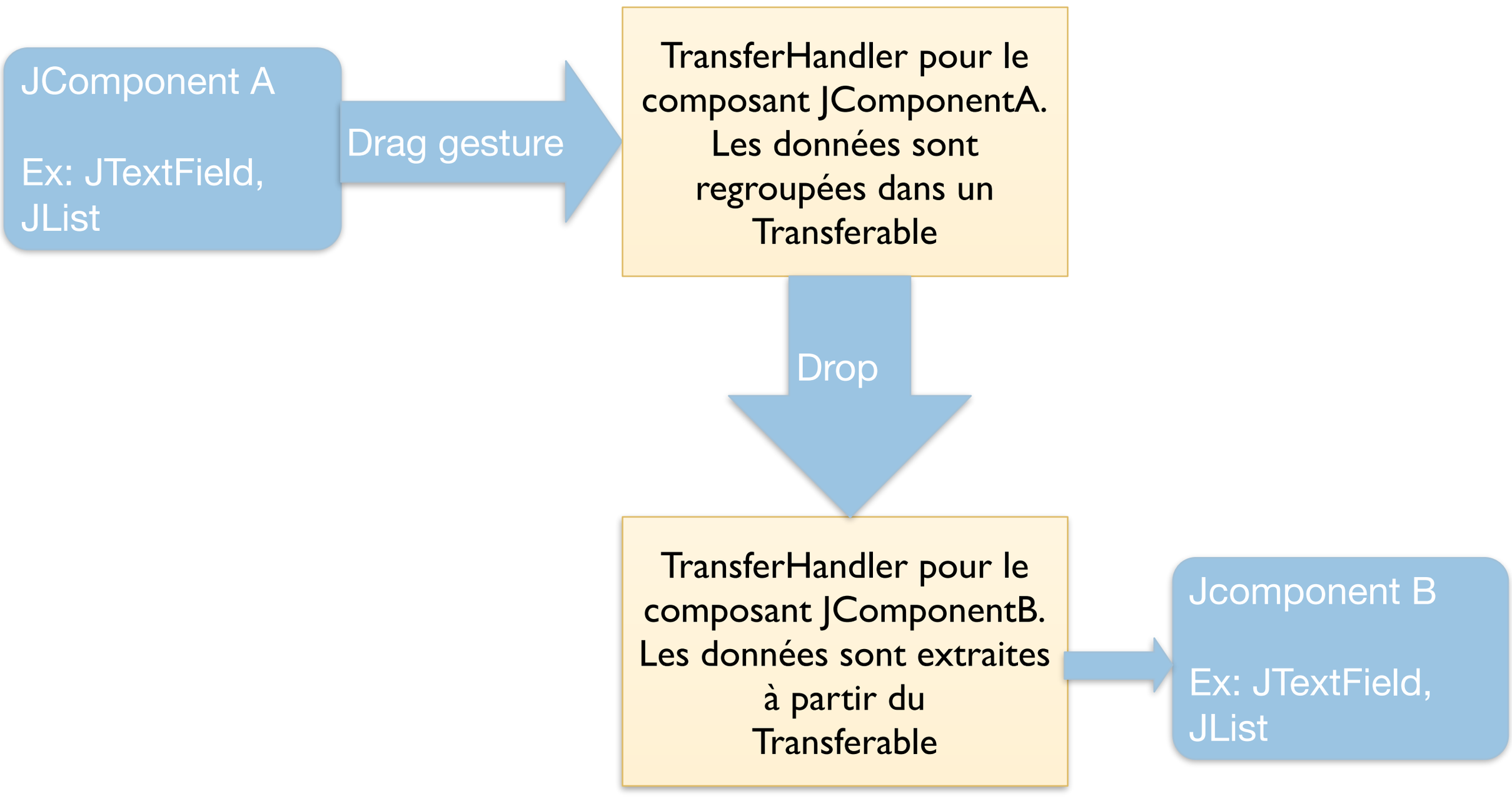
Une donnée transférée

java.awt.dataTransfer.Transferable



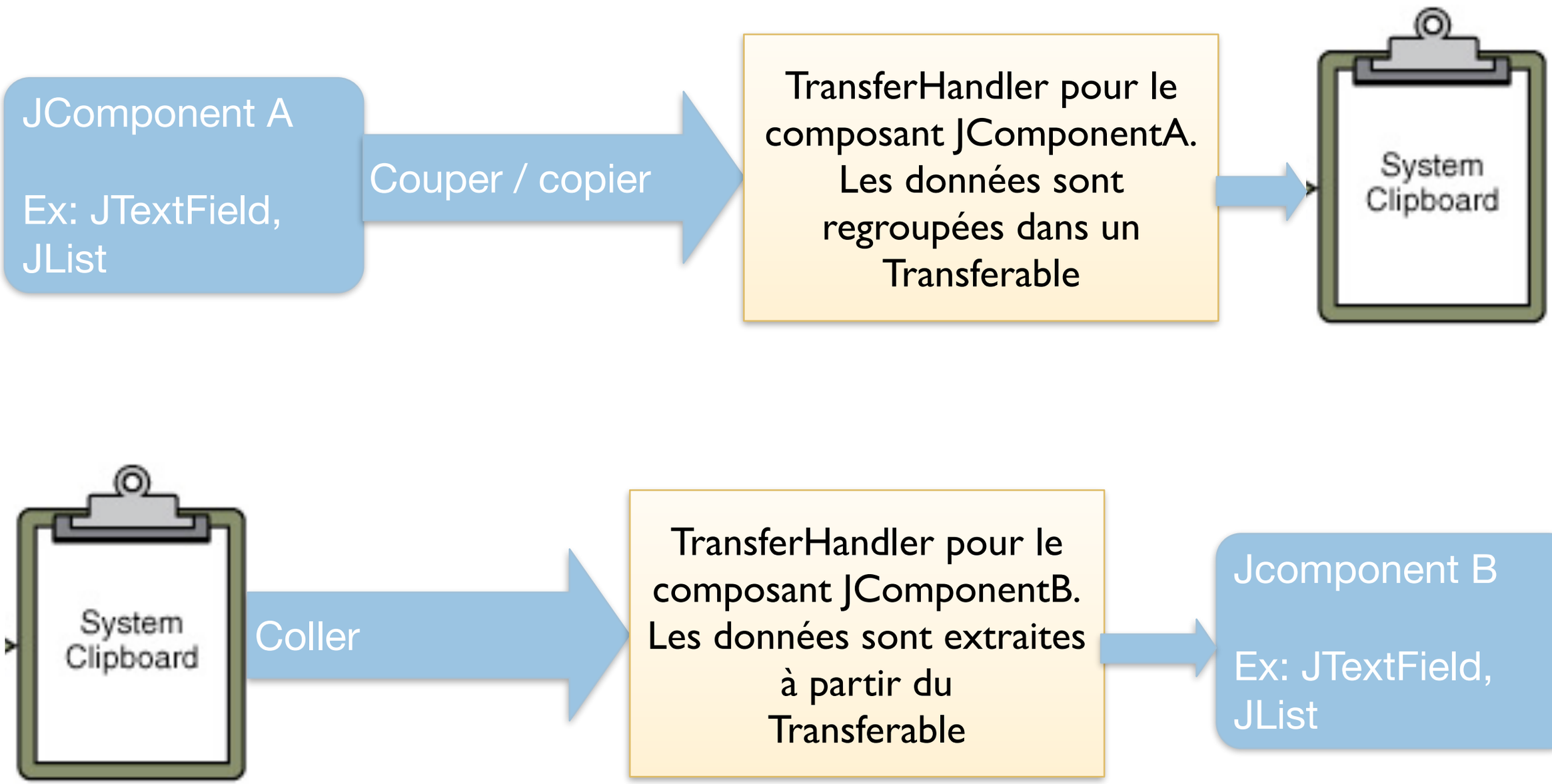
2 façons de transférer des informations

Drag and drop en Java



2 façons de transférer des informations

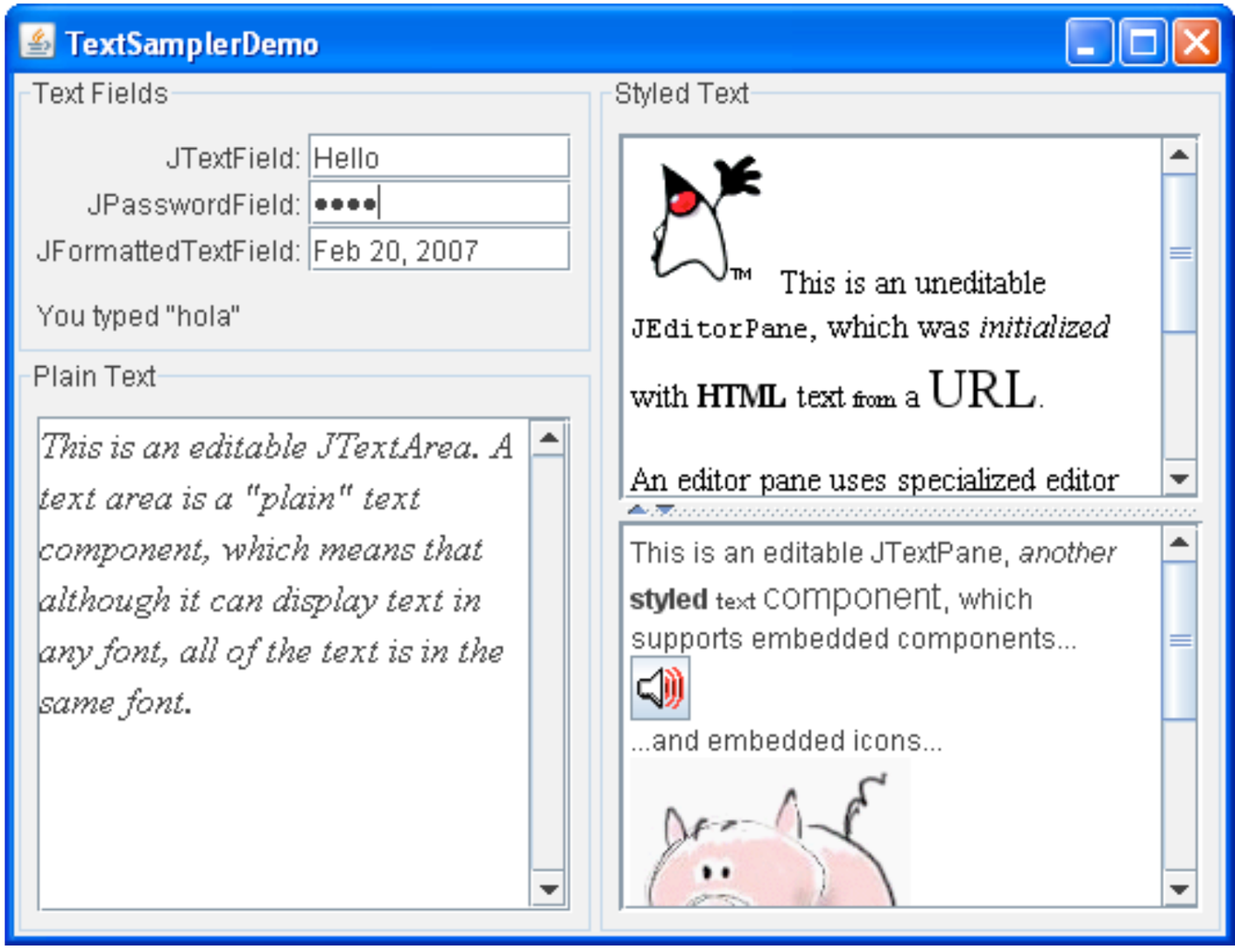
(Couper, Copier)-Coller en Java



DnD sans rien coder

Drop et coller supportés sans écrire une ligne de code (sauf setDragEnable(true)) :

- JEditorPane
- JFormattedTextField
- JPasswordField
- JTextArea
- JTextField
- JTextPane
- JColorChooser

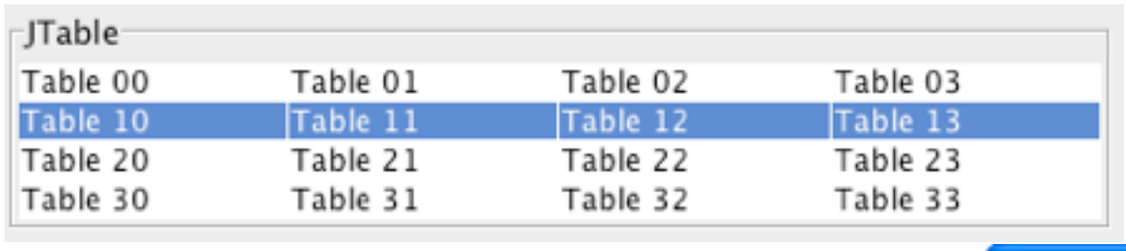


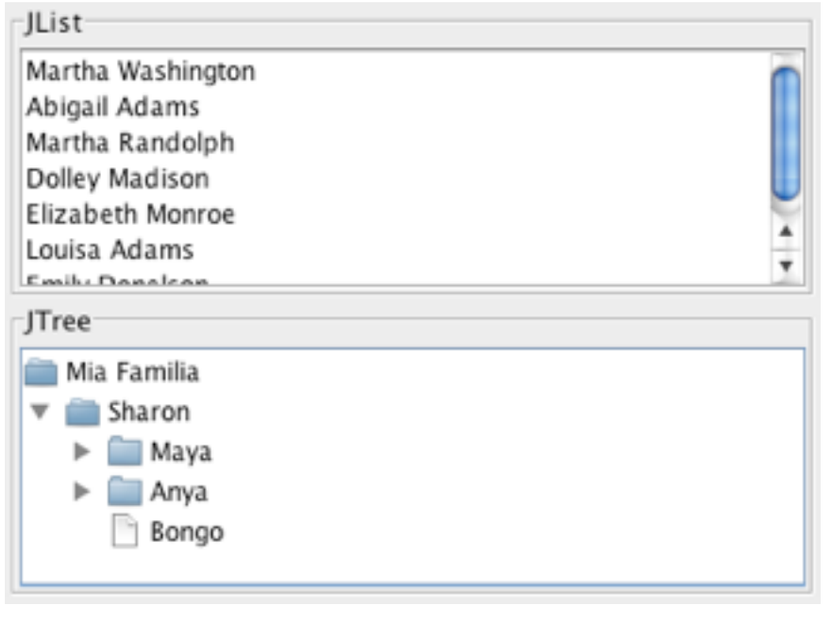
Drag sans rien coder

Pas évident de connaître le comportement par défaut pour tous les composants

setDragEnable(true) + qq lignes de code

- Jlist
- Jtable
- JTree





Support par défaut du DnD

Data Transfer Support

Component	Drag* Copy	Drag* Move	Drop	Cut	Copy	Paste
JColorChooser**	✓		✓			
JEditorPane	✓	✓	✓	✓	✓	✓
JFileChooser***	✓				✓	
JFormattedTextField	✓	✓	✓	✓	✓	✓
JList	✓				✓	
JPasswordField	*n/a*	*n/a*	✓	*n/a*	*n/a*	✓
JTable	✓				✓	
JTextArea	✓	✓	✓	✓	✓	✓
JTextField	✓	✓	✓	✓	✓	✓
JTextPane	✓	✓	✓	✓	✓	✓
JTree	✓				✓	

*Activable en utilisant component.setDragEnabled(true)

** données de type java.awt.Color

*** Export d'une liste de fichiers

La classe TransferHandler

- Possibilité de remplacer l'objet TransferHandler par défaut ou d'ajouter le DnD à un composant qui ne le supporte pas par défaut (ex: le JLabel ne supporte pas le DnD)
- setTransferHandler(TransferHandler): utilise une importation/exportation personnelle des données (JComponent)

Créer son propre TransfertHandler

Méthode 1: Utiliser le constructeur prenant en paramètre une propriété Java Bean à exporter

```
label.setTransferHandler(new TransferHandler("text"));
```

Ne fonctionne que pour des DnD entre composants Swing (ne fonctionne pas avec des applications externes)

Permet d'exporter et d'importer des propriétés (une seule propriété)

```
JLabel label = new JLabel("toto");
label.setTransferHandler(new TransferHandler("font"));
JTextField text = new JTextField("hello");
text.setTransferHandler(new TransferHandler("font"));
text.setFont(new Font("Arial", Font.BOLD, 14));
```

Créer son propre TransferHandler

Méthode 2: spécifier son propre TransferHandler

- setTransferHandler(TransferHandler): utilise une importation/exportation personnelle des données (JComponent)

Exportation de données

- Connaître les actions possibles sur la source (None, Copy, Move, Link ou une combinaison)
  int getSourceActions(JComponent c)
- Prépare les données à transférer
  Transferable createTransferable(JComponent c)
- Termine le travail: nettoyage si nécessaire
  void exportDone(JComponent c, Transferable t, int action)

Fonctionne entre composants Java Swing et avec les applications externes

DataFlavor

Les données glissées sont identifiées par des données de type DataFlavor

Un objet DataFlavor est un objet java qui représente un type MIME comparable à un identifiant (ne stocke pas de données)

Les types MIME permettent aux applications de reconnaître des types de données ex: « text/html », « image/png »

Constantes pour les types les plus courants:

- stringFlavor pour les chaînes de caractères
- imageFlavor pour les images
- javaFileListFlavor pour les transferts de listes de fichiers (fonctionne uniquement entre applications Java)

DataFlavor

Pour créer un DataFlavor, il faut fournir à la construction le type MIME

DataFlavor(String mimeType)

Ex: construction d'un DataFlavor pour la classe Couleur:

```
String mimeType = DataFlavor.javaJVMLocalObjectMimeType +
";class=" + Couleur.class.getName();
try {
        CouleurFlavor = new DataFlavor(mimeType);
       }
        catch (ClassNotFoundException e) {
       return CouleurFlavor;
       }
```

Interface Transferable

L'interface Transferable s'occupe de stocker les données qui sont manipulées par le système de transfert

Method Summary	
Object	**getTransferData**(DataFlavor flavor) Returns an object which represents the data to be transferred.
DataFlavor[]	**getTransferDataFlavors**() Returns an array of DataFlavor objects indicating the flavors the data can be provided in.
boolean	**isDataFlavorSupported**(DataFlavor flavor) Returns whether or not the specified data flavor is supported for this object.

Implémentations par défaut

- StringSelection: transfert de chaines de caractères
- DataHandler: transfert de données de n'importe quel type

Transfert de données: exportation

```
class Transfert extends TransferHandler {
    Transfert() {

    }

    public int getSourceActions(JComponent c) {
        return COPY_OR_MOVE;
    }

    public Transferable createTransferable(JComponent c) {
        return new StringSelection(((JLabel)c).getText());
    }

    public void exportDone(JComponent c, Transferable t, int action) {
        if (action == MOVE) {
            ((JLabel)c).setText("");
        }
    }
}
```

Importation de données

Utilisée pour le drop ou l'action de coller

Vérifier que le composant destination peut accepter les données (JDK 1.6)

- public boolean canImport(TransferHandler.TransferSupport info)

Transfert des données (JDK 1.6)

- public boolean importData(TransferHandler.TransferSupport info)

La classe TransferSupport

- Classe interne de TransferHandler
- Regroupe toutes les informations du transfert
- Connaître le composant destination du transfert (getComponent)
- Méthode de transfert choisie (getDropAction)
- Méthode de transfert choisie par l'utilisateur getUserDropAction
- Méthodes de transfert supportées par la source getSourceDropActions
- Différentes méthodes pour connaître le type de données acceptées par la destination: getDataFlavors

Exemple d'importation

```
public boolean canImport(TransferSupport info) {
    // on importe que les Strings
    if (!info.isDataFlavorSupported(DataFlavor.stringFlavor)) {
        return false;
    }
    return true;
}

public boolean importData(TransferHandler.TransferSupport info) {
    if (!info.isDrop()) {
        return false;
    }

    // Vérifie qu'on a une String flavor
    if (!info.isDataFlavorSupported(DataFlavor.stringFlavor)) {
        System.out.println("Type de donnée pour le drop non acceptée.");
        return false;
    }

    // Récupère le label
    JLabel label = (JLabel) info.getComponent();

    // récupère la chaîne de caractères à dropper.
    Transferable t = info.getTransferable();
    String data;
    try {
        data = (String)t.getTransferData(DataFlavor.stringFlavor);
    }
    catch (Exception e) { return false; }
    //info.setShowDropLocation(true);

    label.setText(label.getText() + data);
    return true;
}
```

DataFlavor personnalisé

```
class Couleur{
    private Integer red = 0;
    private Integer green = 0;
    private Integer blue = 0;
    Couleur(Integer r, Integer g, Integer b) {
            red = r; green = g; blue = b;
    }
    public Integer getR() {
        return red;
    }
    public Integer getG() {
        return green;
    }
    public Integer getB() {
        return blue;
    }
    public String toString() {
        return red + " " + green + " " + blue;
    }
}
```

DataFlavor personnalisé

```
class TransferableCouleur implements Transferable {
    public static DataFlavor couleurFlavor;
    private Couleur c;

    public TransferableCouleur(Integer r, Integer g, Integer b) {
        String mimeType = DataFlavor.javaJVMLocalObjectMimeType + ";class=" + Couleur.class.getName();
        try {
            couleurFlavor = new DataFlavor(mimeType);
            c = new Couleur(r,g,b);
        }catch(ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    public DataFlavor[] getTransferDataFlavors() {
        return new DataFlavor[] {couleurFlavor, DataFlavor.stringFlavor};
    }

    public boolean isDataFlavorSupported(DataFlavor df) {
        return (df.equals(couleurFlavor) || df.equals(DataFlavor.stringFlavor));
    }

    public Object getTransferData(DataFlavor df) throws
    UnsupportedFlavorException, IOException {
        if (df == null) {
            throw new IOException();
        }
        if (df.equals(couleurFlavor)) {
            return c;
        }
        if (df.equals(DataFlavor.stringFlavor)) {
            return c.toString();
        } else {
            throw new UnsupportedFlavorException(df);
        }
    }

}
```

DataFlavor personnalisé

```
class TransfertCouleur extends TransferHandler {
    TransfertCouleur() {
    }

    public int getSourceActions(JComponent c) {
        return COPY;
    }

    public Transferable createTransferable(JComponent c) {
        Color color = c.getForeground();
        return new TransferableCouleur(color.getRed(), color.getGreen(), color.getBlue());
    }

    /*public void exportDone(JComponent c, Transferable t, int action) {
        if (action == MOVE) {
            //to do
        }
    }*/

    public boolean canImport(TransferSupport info) {
        DataFlavor df[] = info.getDataFlavors();
        for (int i=0; i< df.length;i++) {
            if (df[i].equals(TransferableCouleur.couleurFlavor)) {
                return true;
            }
            if (df[i].equals(DataFlavor.stringFlavor)) {
                return true;
            }
        }
        return false;
    }
```

DataFlavor personnalisé

```
public boolean importData(TransferHandler.TransferSupport info) {
    if (!info.isDrop()) {
        return false;
    }

    if (info.isDataFlavorSupported(TransferableCouleur.couleurFlavor)) {
        try{
            Transferable t = info.getTransferable();
            Couleur c = (Couleur) t.getTransferData(TransferableCouleur.couleurFlavor);
            Color color = new Color(c.getR(), c.getG(), c.getB());
            info.getComponent().setForeground(color);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return true;
    } else if (info.isDataFlavorSupported(DataFlavor.stringFlavor)) {
        try{
            Transferable t = info.getTransferable();
            String couleur = (String) t.getTransferData(DataFlavor.stringFlavor);
            Pattern pattern = Pattern.compile("\\d+");
            Matcher matcher = pattern.matcher(couleur);
            Integer r = 0;
            Integer g = 0;
            Integer b = 0;
            if( matcher.find() ) { r = Integer.parseInt(matcher.group()); }
            if( matcher.find() ) { g = Integer.parseInt(matcher.group()); }
            if( matcher.find() ) { b = Integer.parseInt(matcher.group()); }
            Color color = new Color(r, g, b);
            info.getComponent().setForeground(color);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return true;
    }

    return false;
}

}
```

Listener pour le drag

- L’événement a lieu quand le bouton gauche est enfoncé et que la souris se déplace d’au moins 1 pixel
- addMouseMotionListener(MouseMotionAdapter)
- Méthode public void mouseDragged(MouseEvent e)
- exportAsDrag permet d’initier l’export des données

Exemple

```
class DragAndDropListener extends MouseMotionAdapter {

    public void mouseDragged(MouseEvent e) {
        JLabel label= (JLabel) e.getSource();
        label.getTransferHandler().exportAsDrag(label, e, TransferHandler.COPY);
        /*Toolkit toolkit = Toolkit.getDefaultToolkit();
        Clipboard clip = toolkit.getSystemClipboard();
        label.getTransferHandler().exportToClipboard(label, clip, TransferHandler.COPY);*/
        e.consume(); // empêche que l'événement soit transmis à d'autres composants.
    }
}
```

Retour visuel

Visualisation

- Information qui différencie la copie du déplacement
- Visualisation pour représenter les données glissées

Le retour visuel est aussi important pour montrer les possibilités de drop

Montrer les zones où il est possible de faire un drop

Par défaut seules les zones où il est possible de faire un drop fournissent un retour visuel

Possibilité de désactiver le retour visuel ou de toujours l’activer en utilisant setShowDropLocation

(couper,copier) - coller

Le CCP utilise le TransferHandler

```
public boolean canImport(TransferHandler.TransferSupport info) {
    if (info.isDrop()) { // on a un drop
        //...
    } else { // On a un paste
        //...
    }
    //...
```

Ajouter des bindings pour que les actions de CCP sur le handler soient appelées suite aux raccourcis clavier

CCP sur des champs de texte

Sur les composants texte

- Utilisation du DefaultEditorKit qui fournit les actions de CCP
- Mémorise le dernier composant qui avait le focus

```
/**
 * Create an Edit menu to support cut/copy/paste.
 */
public JMenuBar createMenuBar () {
    JMenuItem menuItem = null;
    JMenuBar menuBar = new JMenuBar();
    JMenu mainMenu = new JMenu("Edit");
    mainMenu.setMnemonic(KeyEvent.VK_E);

    menuItem = new JMenuItem(new DefaultEditorKit.CutAction());
    menuItem.setText("Cut");
    menuItem.setMnemonic(KeyEvent.VK_T);
    mainMenu.add(menuItem);

    menuItem = new JMenuItem(new DefaultEditorKit.CopyAction());
    menuItem.setText("Copy");
    menuItem.setMnemonic(KeyEvent.VK_C);
    mainMenu.add(menuItem);

    menuItem = new JMenuItem(new DefaultEditorKit.PasteAction());
    menuItem.setText("Paste");
    menuItem.setMnemonic(KeyEvent.VK_P);
    mainMenu.add(menuItem);

    menuBar.add(mainMenu);
    return menuBar;
}
```

CCP sur des composants quelconques

Utilisation de la classe ActionMap

La classe ActionMap définit des relations entre des Objets

La classe InputMap définit des relations entre des raccourcis claviers et des objets

```
InputMap imap = this.getInputMap();
imap.put(KeyStroke.getKeyStroke("ctrl X"),
    TransferHandler.getCutAction().getValue(Action.NAME));
```

Le presse papier

Il existe plusieurs presse-papiers

- Le presse papier système
  Toolkit toolkit = Toolkit.getDefaultToolkit();
  toolkit.getSystemClipboard();
- Le presse papier de sélection
  Toolkit toolkit = Toolkit.getDefaultToolkit();
  toolkit.getSelectionClipboard();
- Des presses papiers locaux
  new Clipboard(String name);

Le presse papier

Le presse papier contient un même objet sous plusieurs représentations possibles (DataFlavor)

Sous Windows, lorsqu'une application accède au presse papier, aucune autre ne peut y accéder

- Les méthodes d'accès en Java renvoient alors l'exception IllegalStateException

Ergonomie du DnD

Source principale: http://www.ergolab.net/

Apprentissage

Courbes d’apprentissage très courtes: une fois que l’on sait exécuter le geste sous-jacent, il suffit de comprendre que l’on peut glisser – déposer des éléments pour avoir appris

Deux catégories d’objectifs

Lancer une action sur un objet (ex. ajout au panier)

- Dans ce cas déplacement = moyen de lancer une action
- L’objet « référent » reste à son emplacement initial mais il peut être dupliqué symboliquement

Déplacer un objet

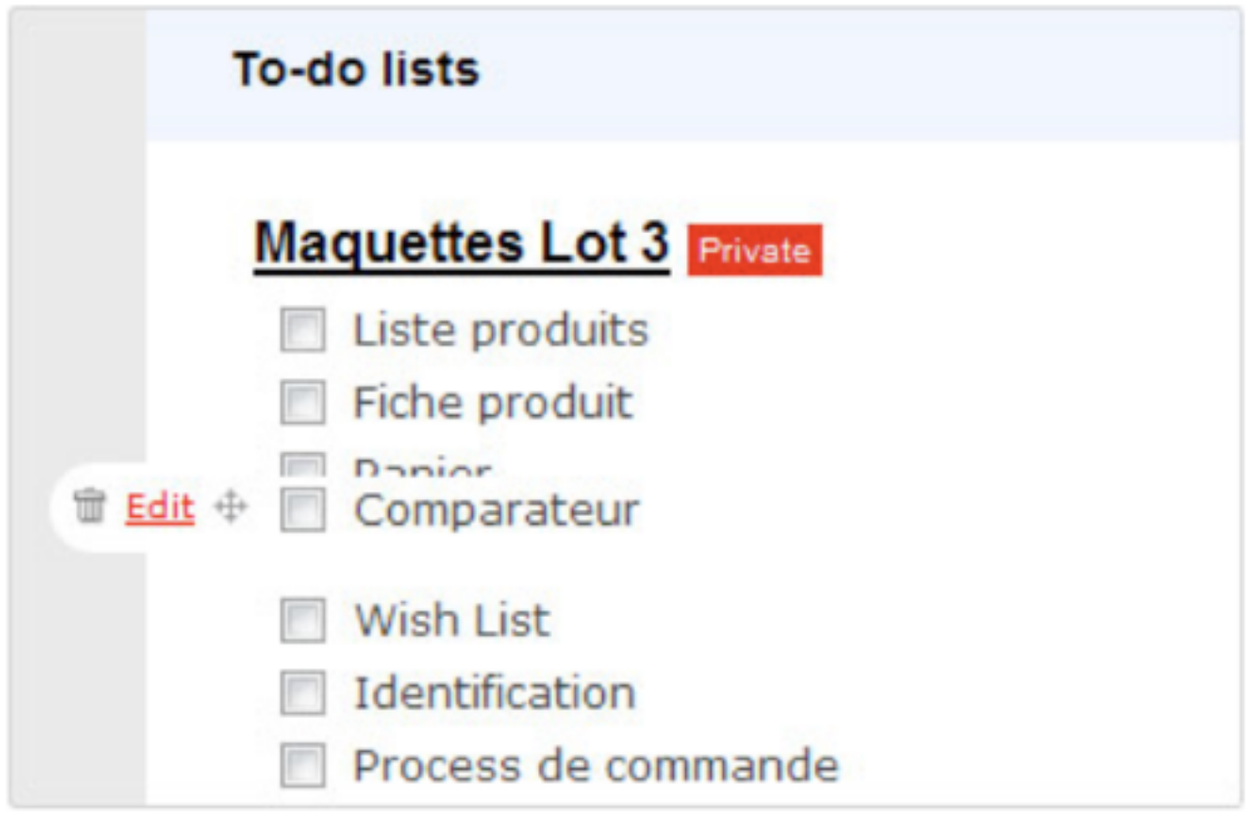
- Dans ce cas, déplacement = finalité
- On a qu’une instance de chaque objet

Deux catégories d’objectifs

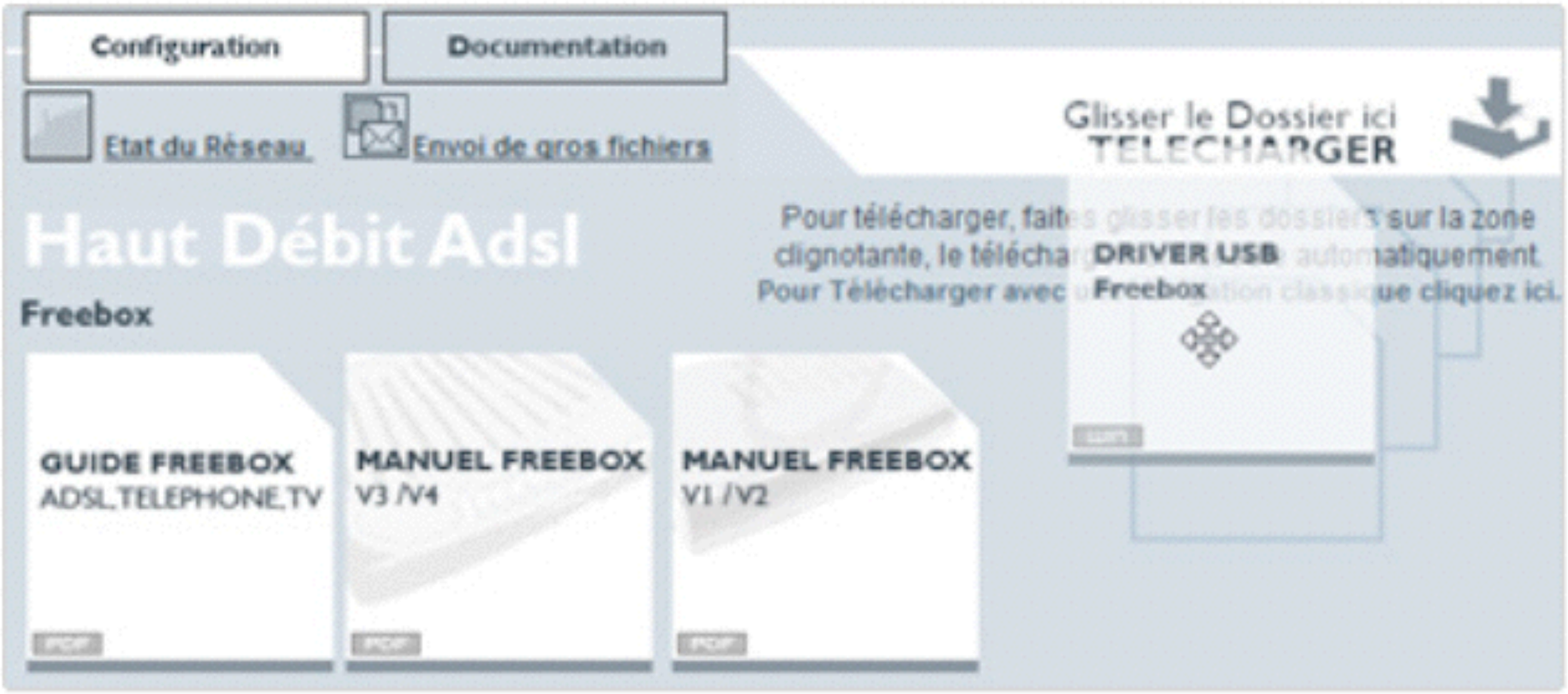
Déplacer un objet

Exemple: réorganiser une liste

Permet de réaliser plus simplement et rapidement l’objectif

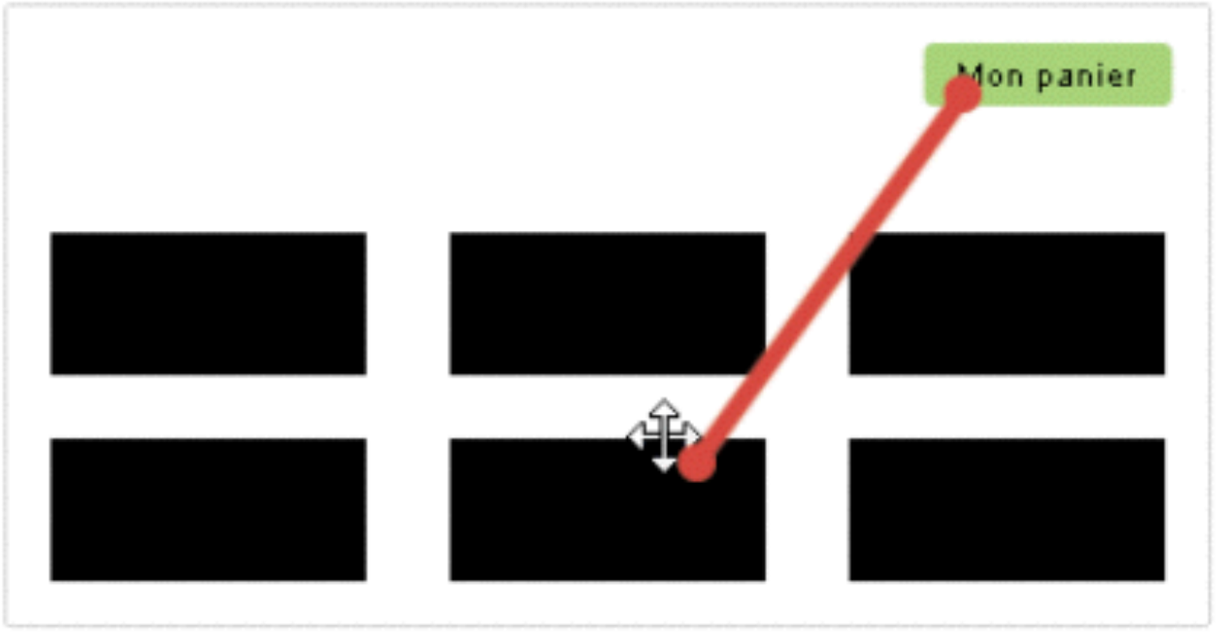


Exemple négatif



Exemple négatif

Ajout des éléments 1 par 1 au panier



La possibilité de DnD doit être visible

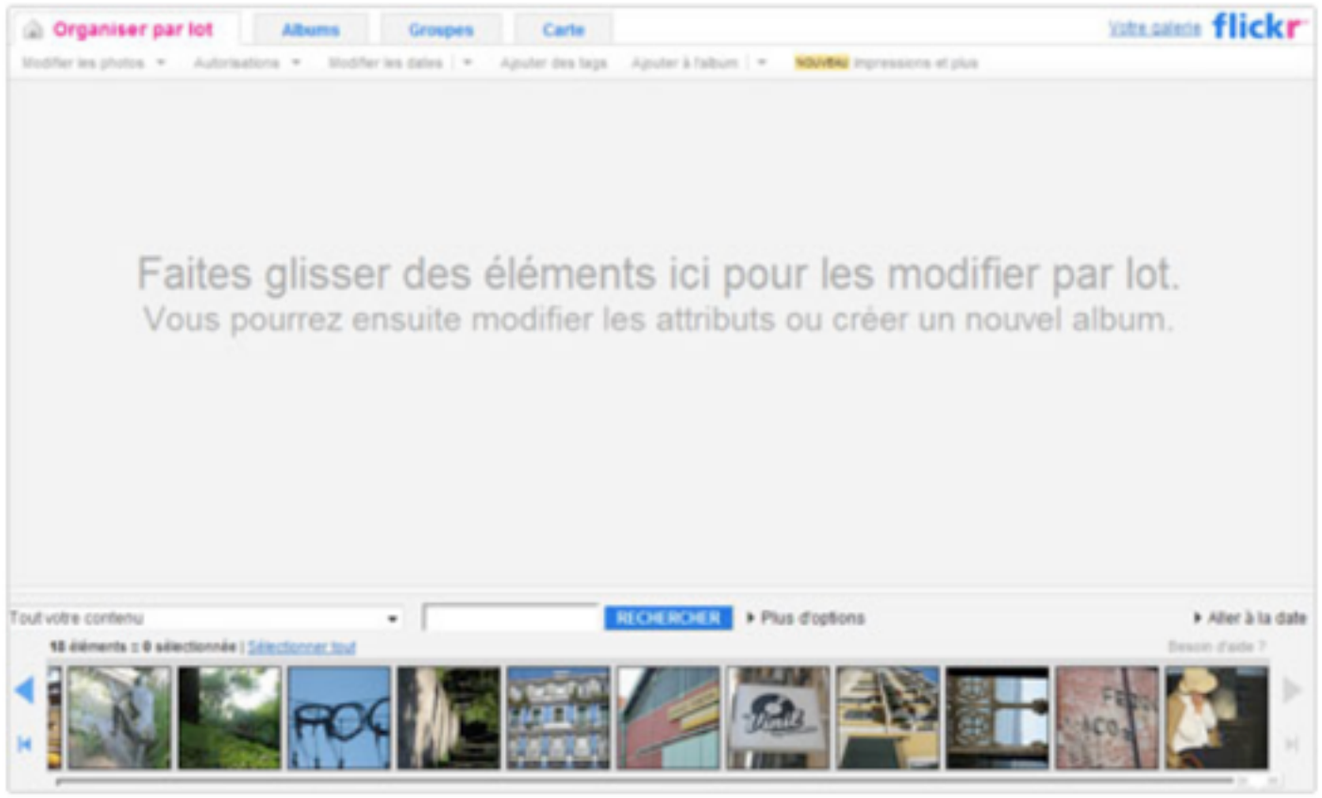
Besoin de deux connaissances pour utiliser le DnD:

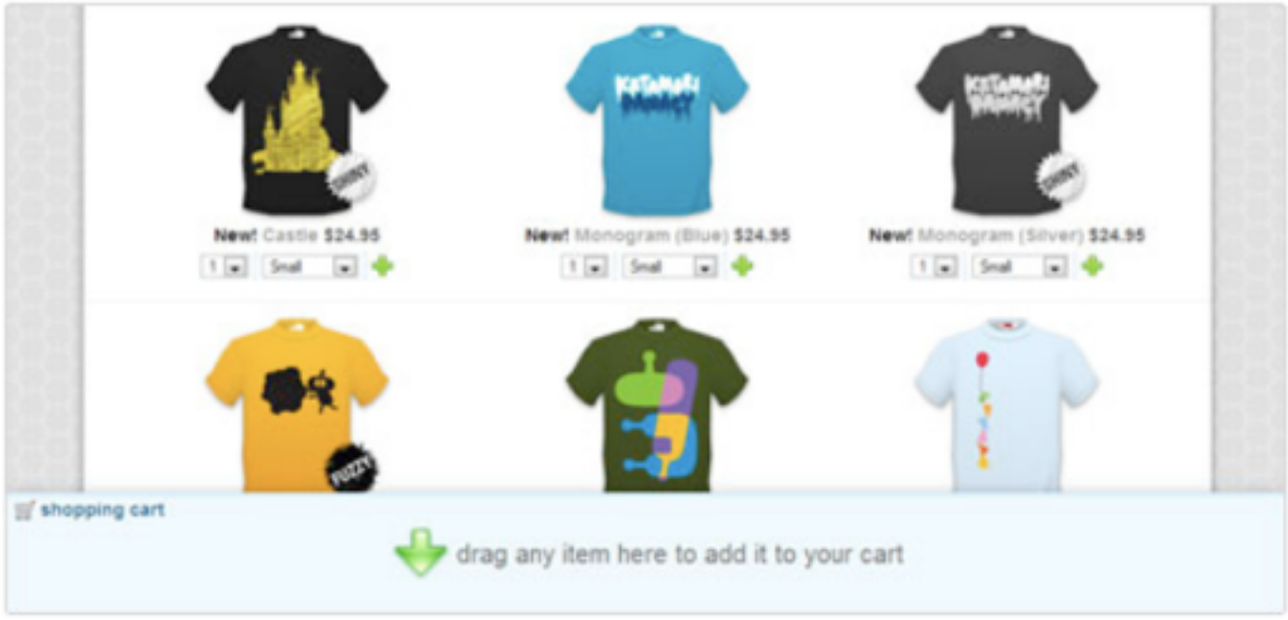
- Savoir qu'il est possible de glisser-déposer des objets
- Savoir quels objets peuvent être glissés déposés

Jouer sur:

- les formats de présentation
- Terminologies
- Comportements au survol

Message adossé au récepteur

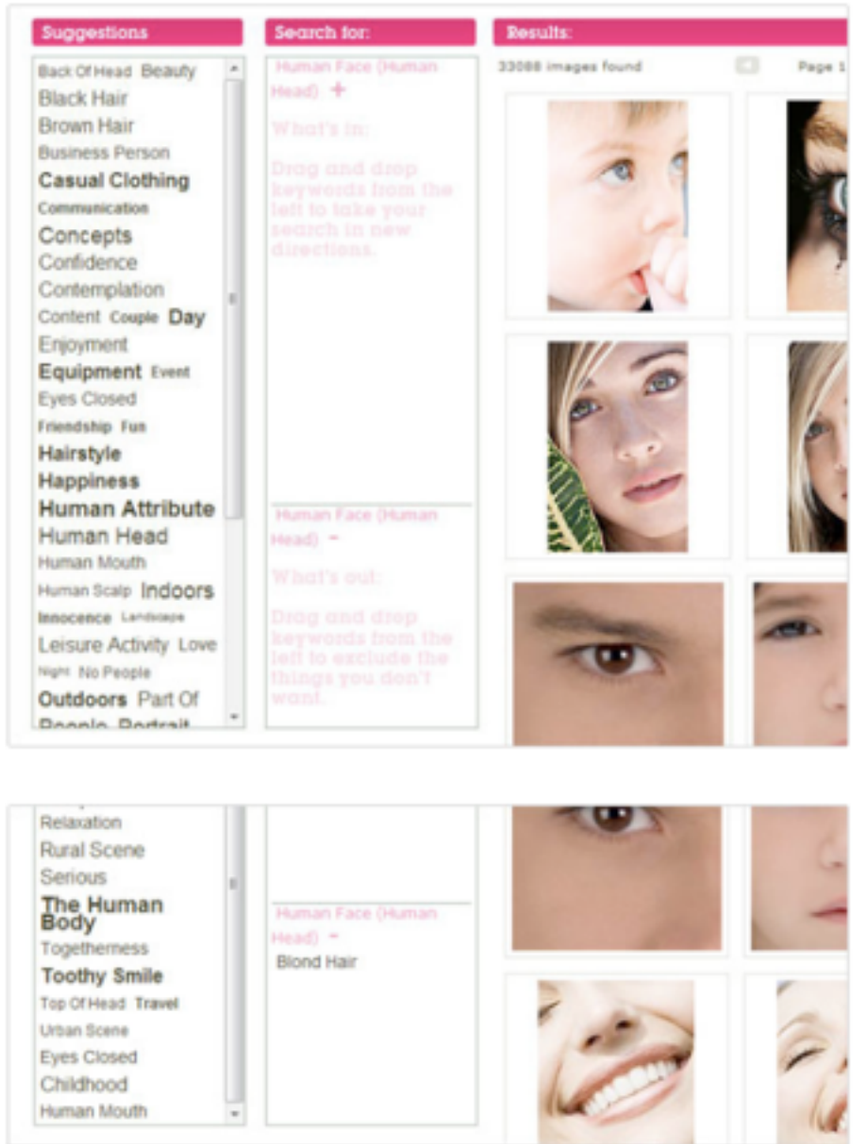




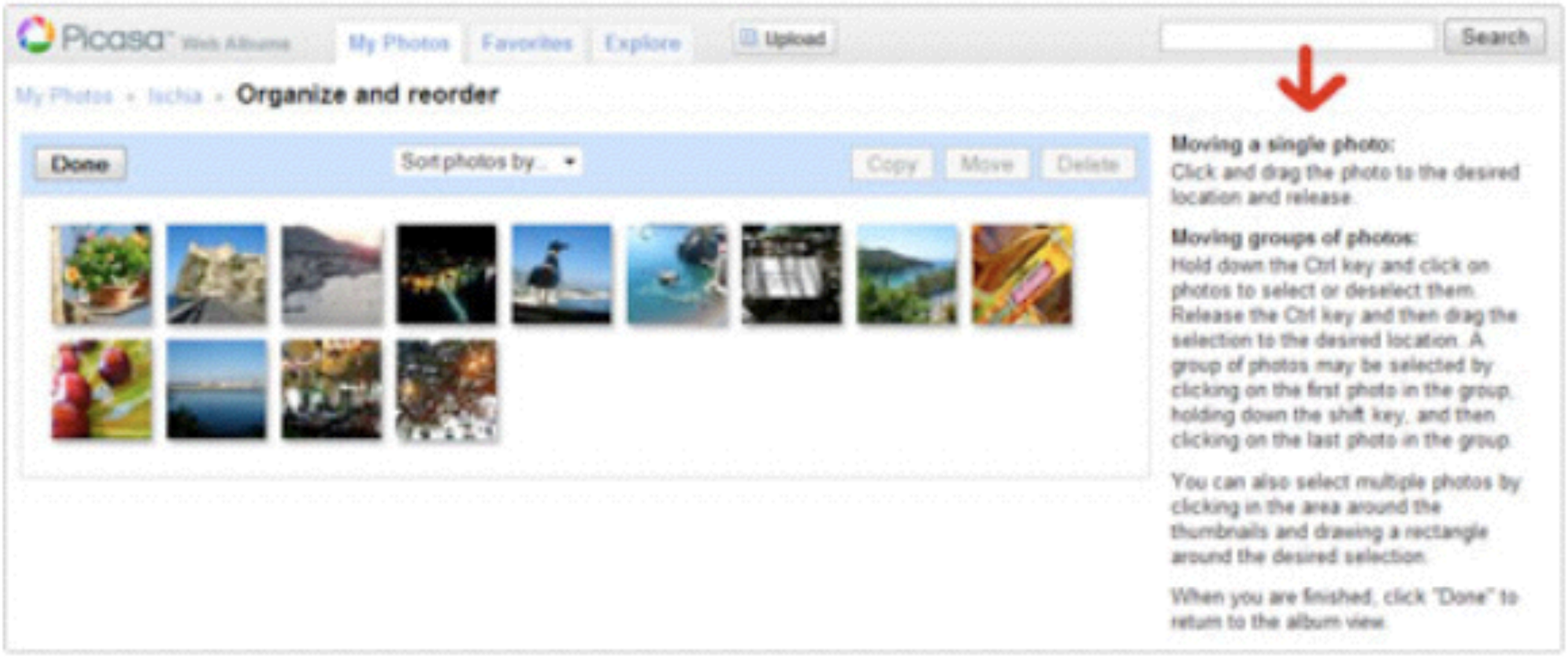
Adaptation du message suivant l’importance de la fonction

F

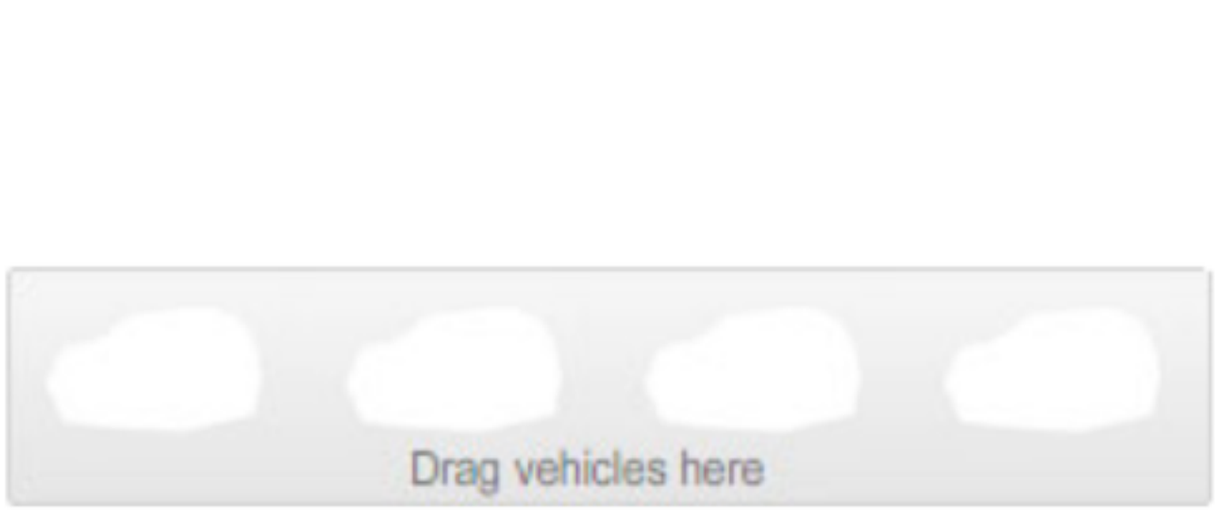
Adaptation du message suivant l’importance de la fonction

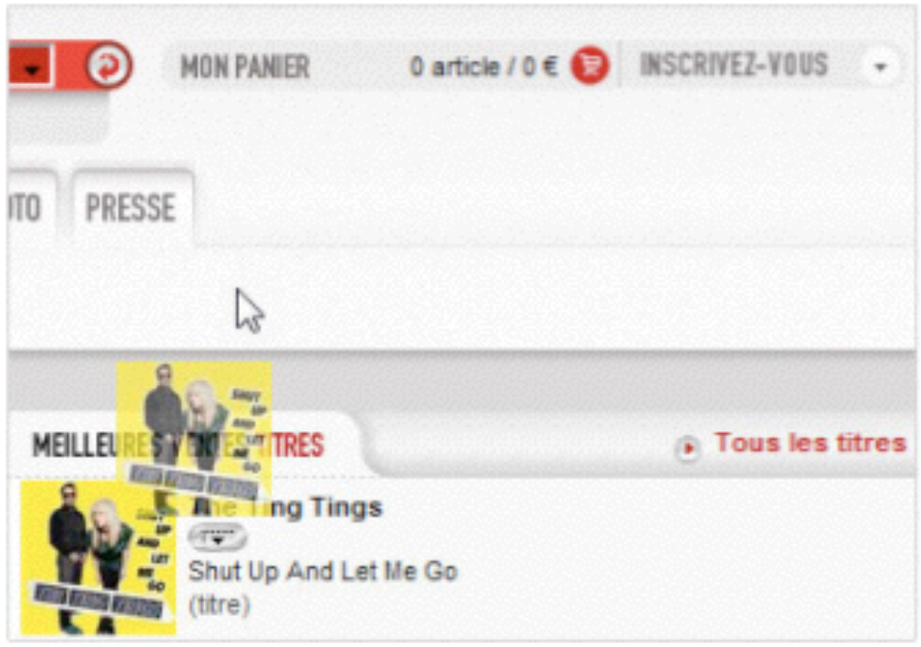


Aide permanente



La destination doit être visible





Indiquer la possibilité plutôt que l’interdiction

Il vaut mieux indiquer où un objet est déposable, plutô

Paramètre influençant l’efficacité

La destination est d’autant plus facile à atteindre qu’elle est proche et grande

Plus la source est de taille importante, plus elle est facile à prendre

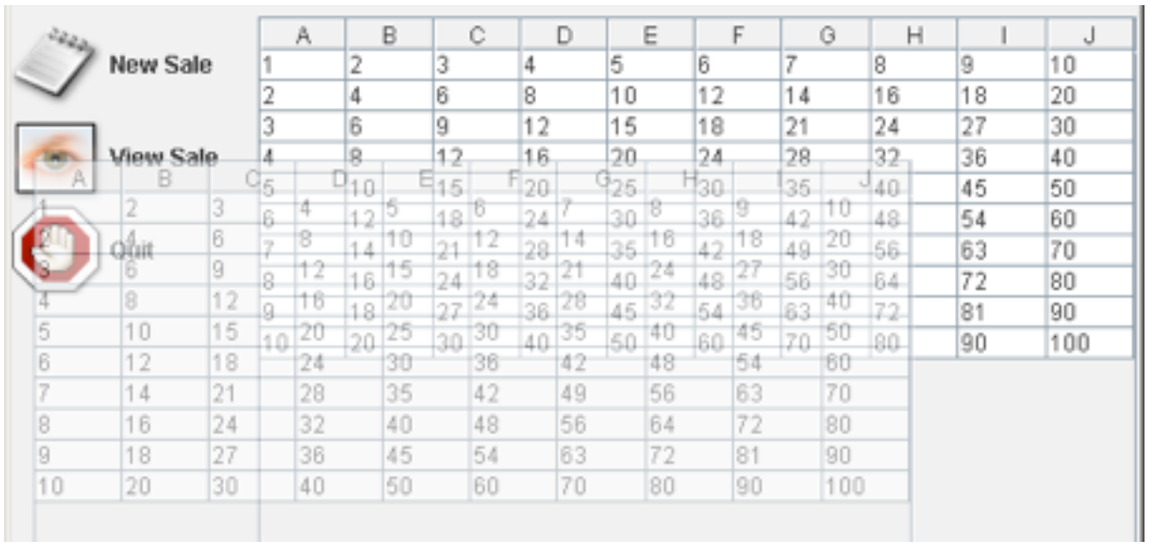
La visibilité simultanée de la source et de la destination

Le déplacement doit être visible

Fournir un retour d’information informant l’utilisateur sur le fait qu’il est en train de déplacer un élément

- Eviter des erreurs du type: « je crois déplacer un objet que j’ai mal saisi, donc je ne déplace rien »
- Retour en temps réel sur l’efficacité de son action de déplacement
- Contrôler avec précision son action en vue de la dépose de l’objet à sa destination

Le déplacement doit être visible



Problèmes liés au DnD

Demande plus d'effort physique que de déplacer la souris sans maintenir de bouton enfoncé (donc moins rapide)

Un usage intensif peut être cause de douleurs

Problème quand la cible est cachée sous d'autres objets

- Utilisation d'Exposé sous Mac OS X

Et aussi…

DnD récurssif

- Naviguer dans une arborescence en restant quelques secondes sur un niveau pour pénétrer au niveau d’arborescence inférieur

Le pick and drop

1997

Utilisation d’un stylo pour prendre des données d’un ordinateur à un autre

Drag and pop

Affichage des cibles potentielles suivant la direction du mouvement

Seules les cibles compatibles sont affichées

Fold and drop