Fondements des systèmes graphiques interactifs

Sylvain Malacria

http://www.malacria.com/

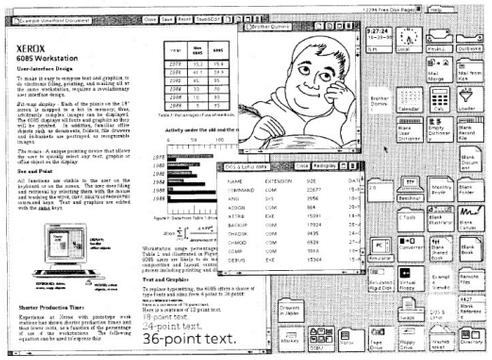
mailto:sylvain.malacria@inria.fr

Diapositives adaptées de Nicolas Roussel

Objectifs

Discuter de manière non exhaustive des principes fondamentaux des systèmes graphiques interactifs actuels

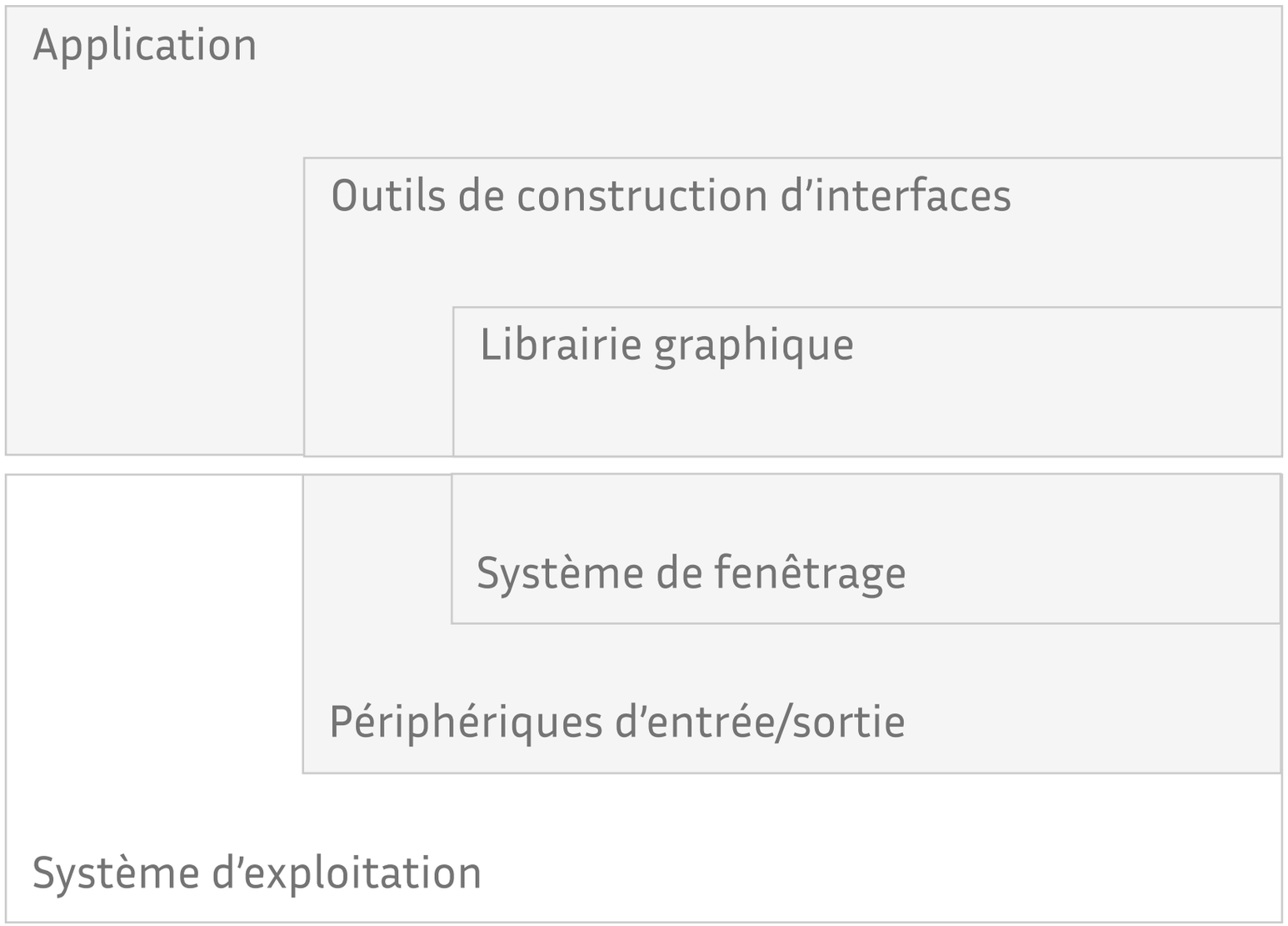
Point de départ : le Xerox Star et les systèmes qui s'en sont inspirés

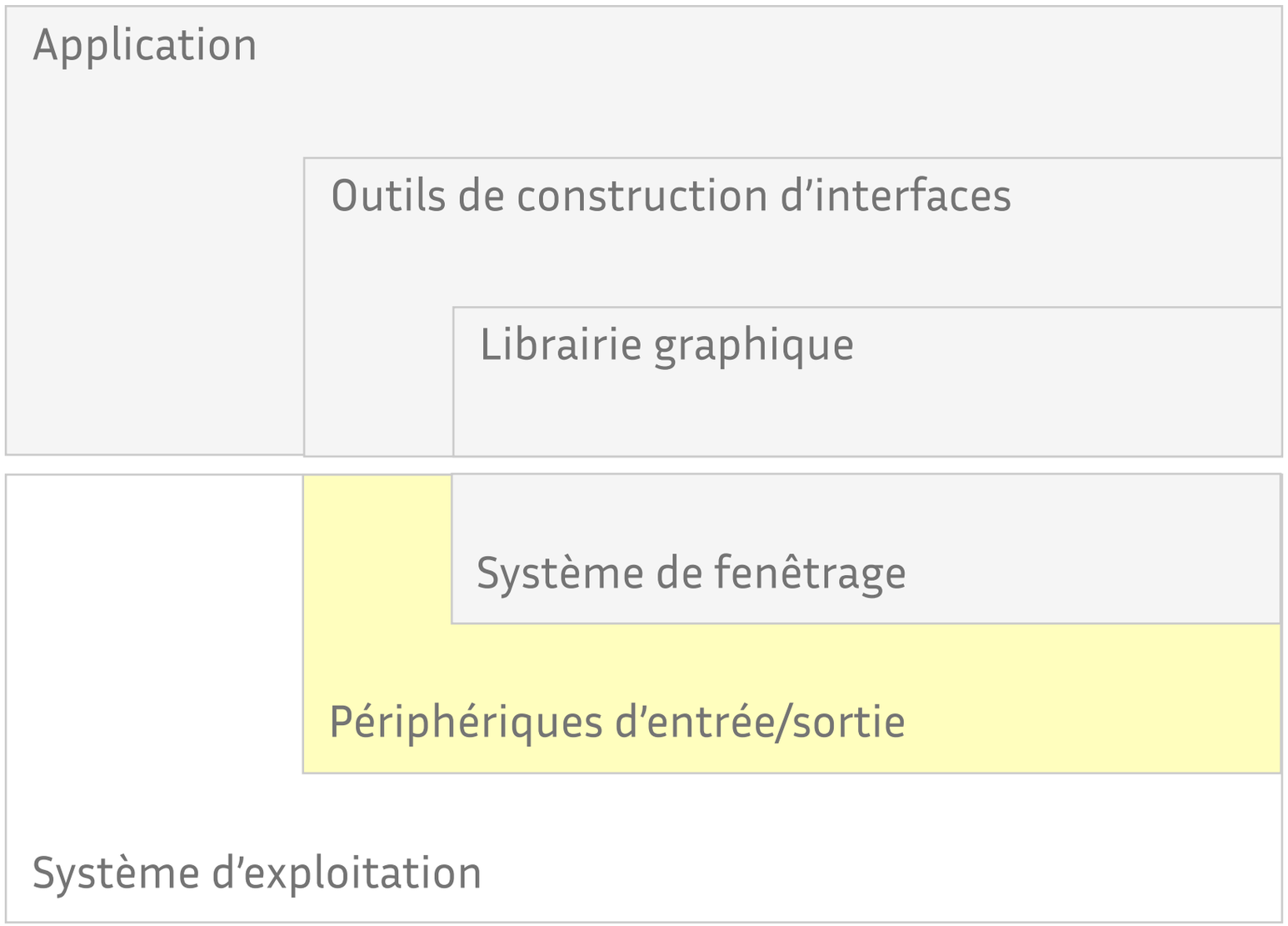


Ces systèmes sont encore largement utilisés et leurs principes se retrouvent dans les systèmes plus récents

"*If you are going to break something, including a tradition, the more you understand it, the better job you can do*" (W. Buxton, 2007)

Architecture générale d'un système graphique interactif




Application
Outils de construction d’interfaces
Librairie graphique
Système de fenêtrage
Périphériques d’entrée/sortie
Système d’exploitation

Modalités et périphériques d'entrée : quelques exemples

Claviers, boîtes à boutons

Potentiomètres (rotatifs, linéaires)

Souris, tablettes, joystick, trackball

Crayons optiques

Surfaces tactiles

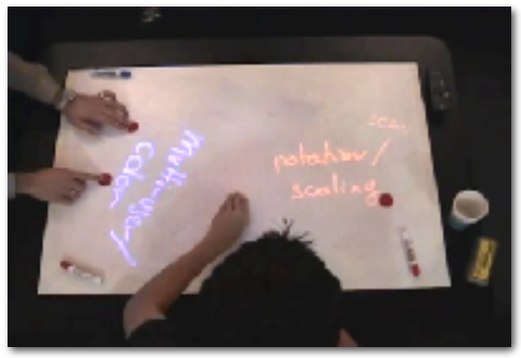
Accéléromètres, gyroscopes, etc

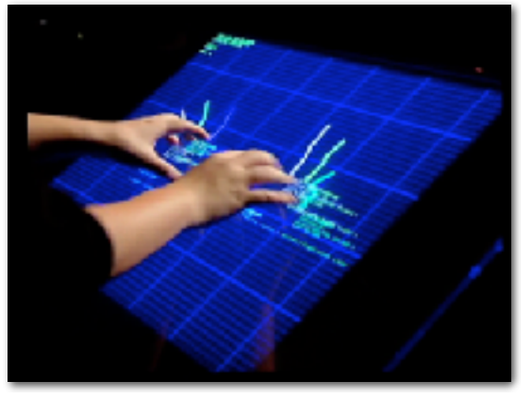
Capteurs de position et direction

Sons, parole

Vision par ordinateur

Interfaces cerveau-ordinateur





Du périphérique physique au périphérique logique

Exemple : périphériques donnant une information de position

- localisation absolue (tablette, écran tactile, crayon optique)
  ou relative (souris, joystick, trackball)
- interaction directe (écran tactile, stylo optique)
  ou indirecte (souris, joystick, trackball, tablette ?)

Le passage du monde physique au monde logique se fait via une *fonction de transfert* (nous y reviendrons...)

Gestion des périphériques d'entrée

Le système peut interroger le périphérique (*polling*)

- de manière bloquante (mais on ne peut rien faire en attendant...)
- de manière non bloquante (mais on risque d'interroger souvent pour rien)

L'envoi de données par le périphérique peut aussi déclencher une *interruption* permettant au système de mettre ces données dans une file d'*événements* à traiter ultérieurement

Le périphérique peut avoir très souvent des choses à dire...

- 50, 60 fois par seconde ou plus pour des caméras
- 100 à 1000 fois par seconde pour une souris, un accéléromètre, etc
- des dizaines ou centaines de milliers de fois par seconde pour un micro

Un périphérique intéressant : la Wiimote

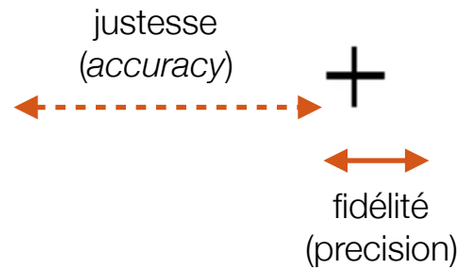
1 pad directionnel + 8 boutons
1 accéléromètre 3 axes
1 caméra sensible à l'infrarouge

4 LEDs
1 vibreur
1 haut-parleur

1 liaison BlueTooth
1 connecteur pour extension (e.g. Motion Plus pour gyroscope)

Pour les détails techniques, voir http://wiibrew.org/wiki/Wiimote

Les données sont souvent bruitées…



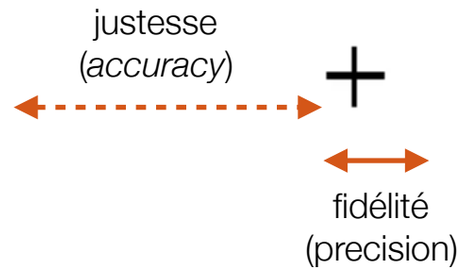
La moyenne glissante n’est pas une solution

Un filtre passe bas du premier ordre est meilleur, mais le compromis précision/réponse reste difficile à trouver

Une solution efficace et pas chère : à l’aide d’un premier filtre passe bas, estimer la dérivée et s’en servir pour déterminer la fréquence de coupure d’un deuxième filtre passe bas appliqué sur les données (on y reviendra…)

http://www.lifl.fr/~casiez/1euro/

Les données sont souvent bruitées…



La moyenne glissante n’est pas une solution

Un filtre passe bas du premier ordre est meilleur, mais le compromis précision/réponse reste difficile à trouver

Une solution efficace et pas chère : à l’aide d’un premier filtre passe bas, estimer la dérivée et s’en servir pour déterminer la fréquence de coupure d’un deuxième filtre passe bas appliqué sur les données (on y reviendra…)

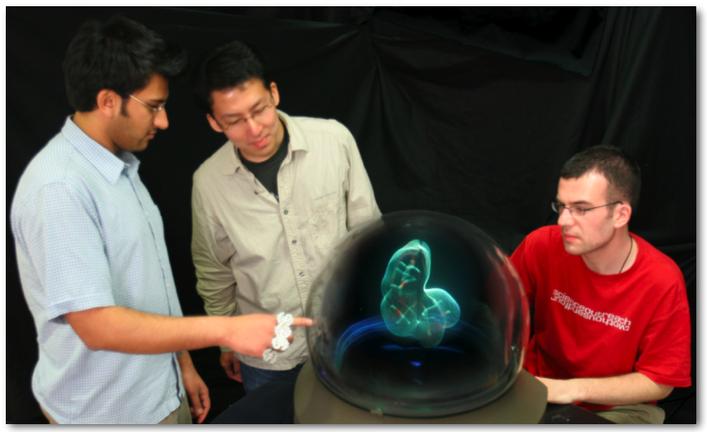
http://www.lifl.fr/~casiez/1euro/

Périphériques d'affichage : quelques exemples

Apple Macintosh 128k
512 x 342 pixels, 9”
175 104 pixels au total

Apple iPhone 5S
640 x 1136 pixels, 4”
727 040 pixels au total

Ecran Apple Thunderbolt
2560 x 1440 pixels, 27”, 12 ms
3 686 400 pixels au total



Ecran volumétrique
192 x 768 x 768 pixels, 10“
11 3246 208 pixels au total

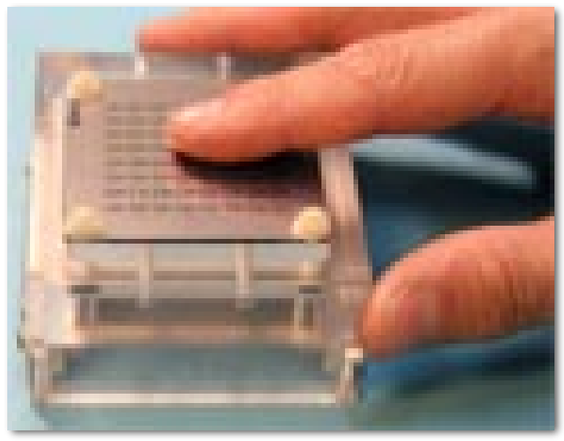
HiPerSpace, UC San Diego
14 x 5 écrans 30” de 2560 x 1600 pixels
286 720 000 pixels au total (35840 x 8000)

Autres modalités et périphériques de sortie

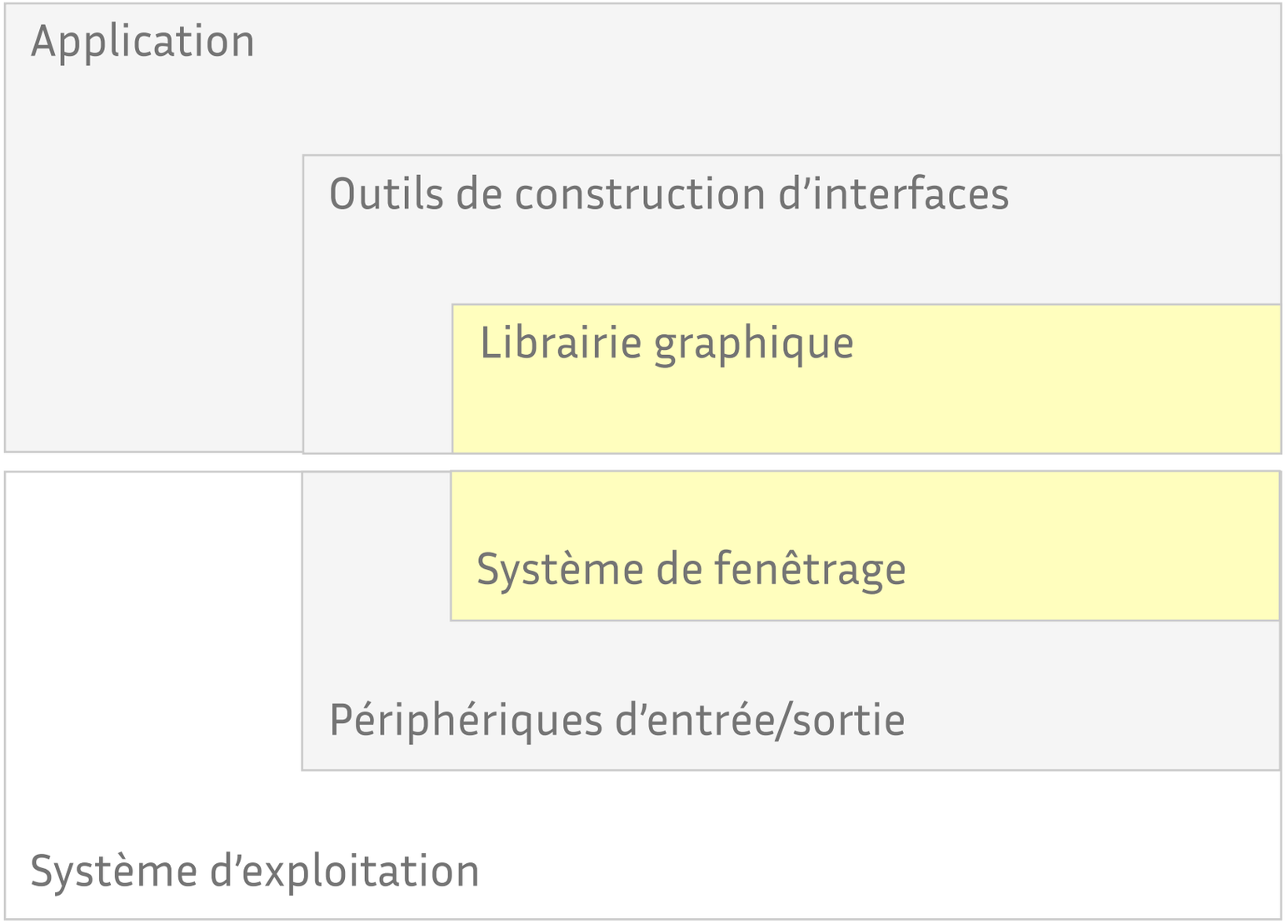
Diffuseur d’odeur (Exhalia)

STIMTAC
(L2EP, Univ. Lille 1)

Geomagic Touch X
6DOF, 7.9 N, > 1100 dpi



Vital
(CEA LIST)


Application
Outils de construction d’interfaces
Librairie graphique
Système de fenêtrage
Périphériques d’entrée/sortie
Système d’exploitation

Système de fenêtrage

Le système de fenêtrage (ou système graphique) permet aux applications d'utiliser les différents périphériques d'entrée/sortie

Une fenêtre est une zone autonome pour l'affichage et/ou l'entrée de données

Le système de fenêtrage s'occupe de

- l'ouverture et la fermeture d'applications, la gestion de session
- la gestion des fenêtres
- l'affectation des périphériques d'entrée (*focus*)
- la communication entre applications (e.g. copier-coller, glisser-déposer)

Dessiner dans une fenêtre : quel modèle ?

Eléments constituants d'un modèle graphique

- un ensemble de primitives graphiques
- un ensembles d'attributs sur ces primitives
- une sémantique graphique et des fonctions associées (e.g. clipping)

Principaux modèles utilisés : pixellaire, vectoriel 2D, vectoriel 3D

Le dessin peut être direct (modèle du peintre) ou structuré (graphe d'objets, graphe de scène)

Le modèle est mis en œuvre dans une ou plusieurs librairie(s)

Exemples de librairies/modèles : Xlib, Java2D, PDF, SVG, OpenGL

Modèle pixellaire + vectoriel 2D

A chaque pixel sont associées trois composantes de couleur (rouge, vert, bleu) et une composante d'opacité (alpha)

Les pixels sont produits par des primitives de dessin

- lignes droites ou courbes (splines, bezier)
- formes libres
- polygones (triangles, rectangles, etc)
- arcs, cercles, ellipses
- texte (police+taille+style)
- dessin bitmap

Dessiner dans une fenêtre : quelles coordonnées ?

Plusieurs systèmes de coordonnées co-existent

Celui lié aux objets visualisés

- dont les unités dépendent des objets considérés
- indépendant de l'application

Celui lié à l'adressage de l'écran

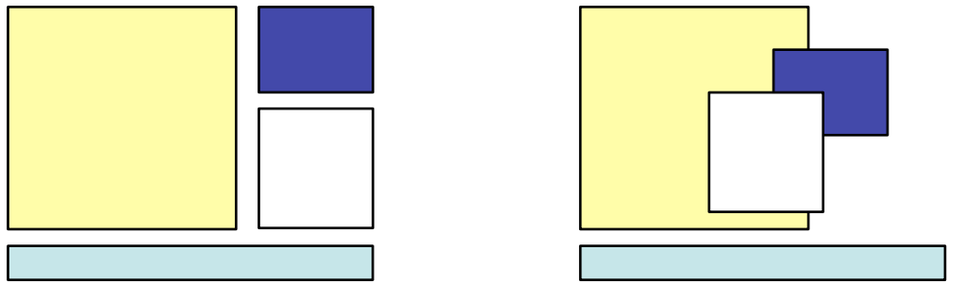
- en pixels
- quelle origine (e.g. écran(s) ou fenêtre) ? quelle orientation ?

Le monde réel

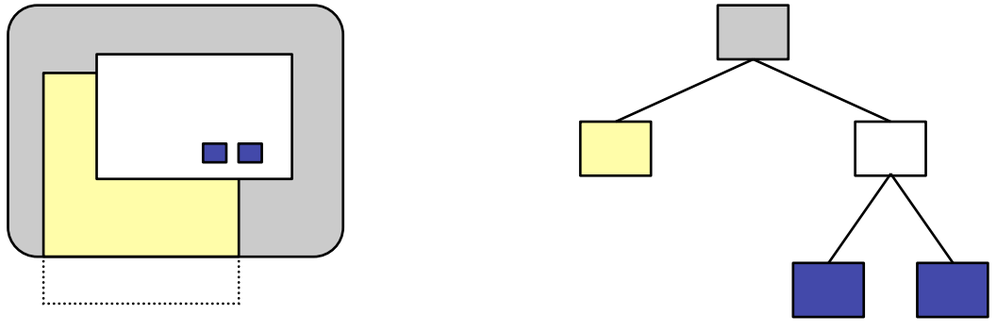
- en mètres (ou décimètres, centimètres, millimètres)
- indépendant du périphérique mais lié à la visualisation choisie

Dessiner plusieurs fenêtres : quel modèle ?

Avec ou sans recouvrement



Eventuellement hiérarchique



En cas de recouvrement, le système doit pouvoir réafficher les parties cachées (nécessite de mémoriser le contenu des fenêtres) ou demander aux applications de le faire (les mécanismes de clipping peuvent être utiles)

Interagir avec le contenu d'une fenêtre

Une fenêtre est un terminal virtuel associé à une application

L'application reçoit (si elle le souhaite) les événements qui se produisent dans toutes ses fenêtres

Suivant le périphérique, différentes gestions du *focus* sont possibles

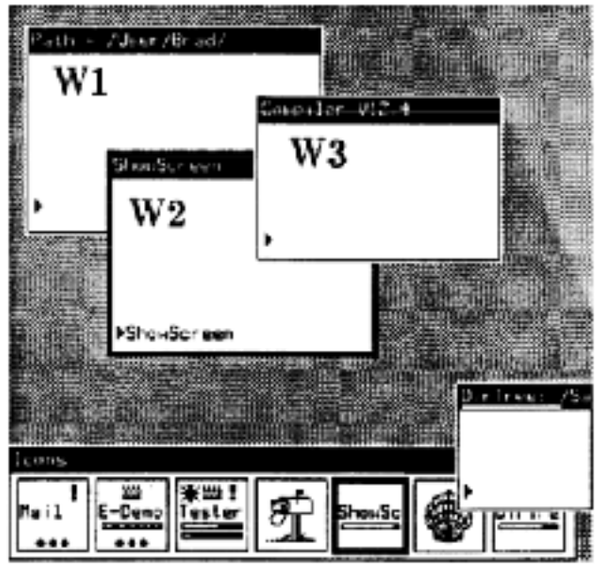
- explicite : les événements sont envoyés à une fenêtre particulière spécifiée antérieurement
- implicite : les événements sont envoyés à la fenêtre correspondant à la position (x,y) qu'ils contiennent
- esclave : le focus suit celui d'un autre périphérique

Par défaut, s'ils n'ont pas été traités par une application, les événements sont traités par le système de fenêtrage

Manipuler/gérer les fenêtres

"A window manager is a software package that helps the user monitor and control different contexts by separating them physically onto different parts of one or more display screens (...) Before window managers, people had to remember their various activities and how to switch back and forth"

B. Myers. "A taxonomy of window manager user interfaces"
IEEE Computer Graphics and Applications 8(5), 1988

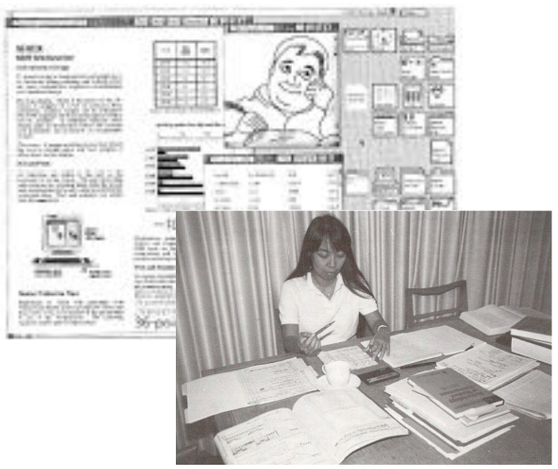


De très nombreuses opérations, via des décorations (cadre et barre de titre), des menus, icônes et raccourcis clavier

- placer et déplacer, dimensionner
- iconifier, maximiser, restaurer la taille initiale
- réduire à la barre de titre, cacher et révéler
- toujours au-dessus ou en-dessous
- fermer

Quelles évolutions en 30 ans ?

Xerox Star, 1981



Apple MacOS 1.1, 1984

Microsoft Windows 2.03, 1987

X Window + TWM, 1989

Apple OS X 10.12

Microsoft Windows 10

Ubuntu

Quelles évolutions en 30 ans ?

Les technologies d'entrée/sortie ont grandement évolué

Les usages des systèmes graphiques interactifs se sont multipliés

Certains aspects de leur architecture ont profondément changé, d'autres sont restés rigoureusement identiques

La recherche en Interaction Homme-Machine a proposé de nombreuses alternatives... qui restent souvent méconnues

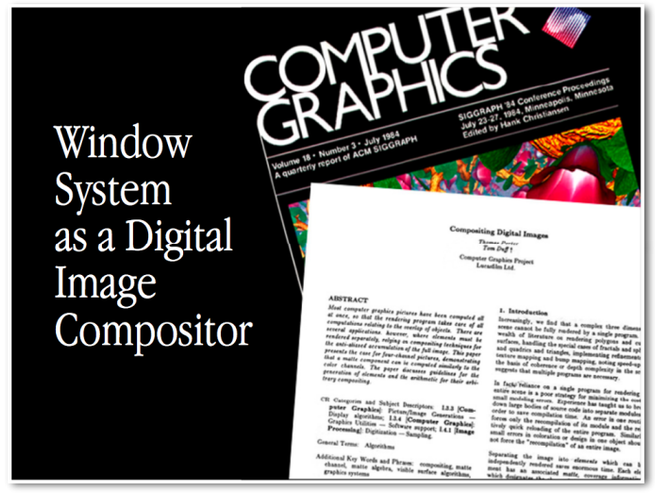
Les applications ne dessinent plus (directement) à l'écran

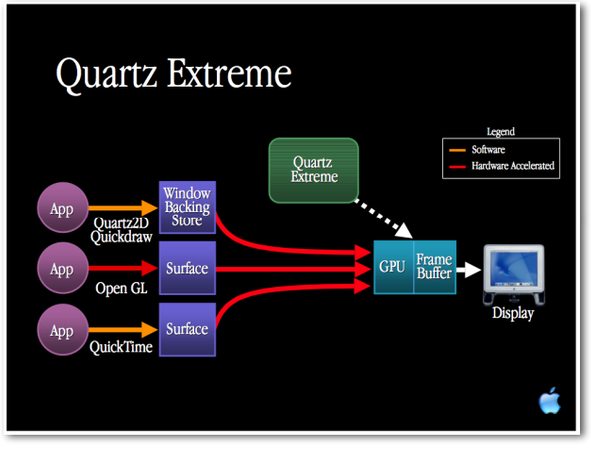
Constat d'Apple en 2001 : le GPU évolue plus vite que le CPU

- CPU : les performances doublent tous les 18 mois (loi de Moore)
- GPU : les performances doublent tous les 6 mois

Quartz Extreme

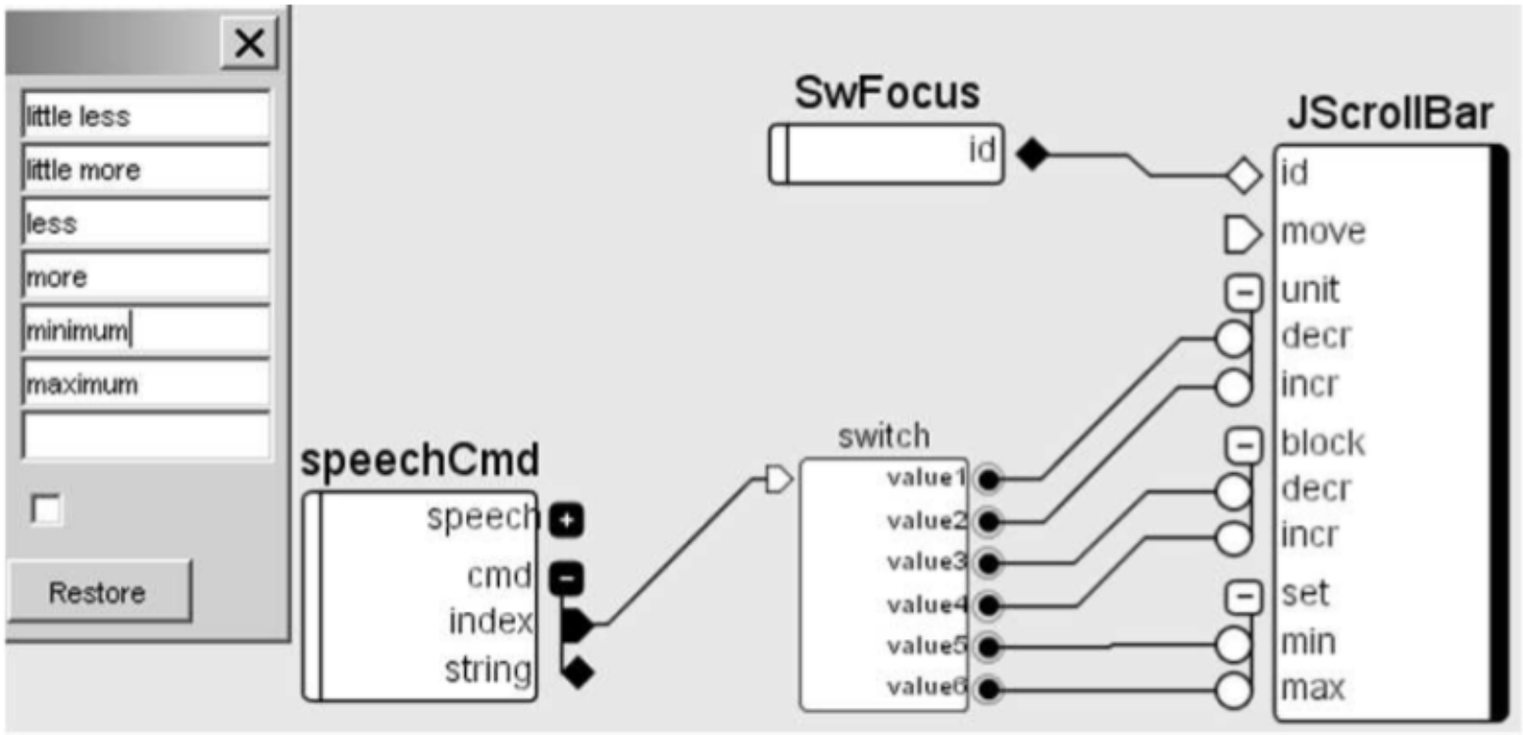
- *"window system as a digital image compositor"*
- de nombreux effets ("*because we can*")



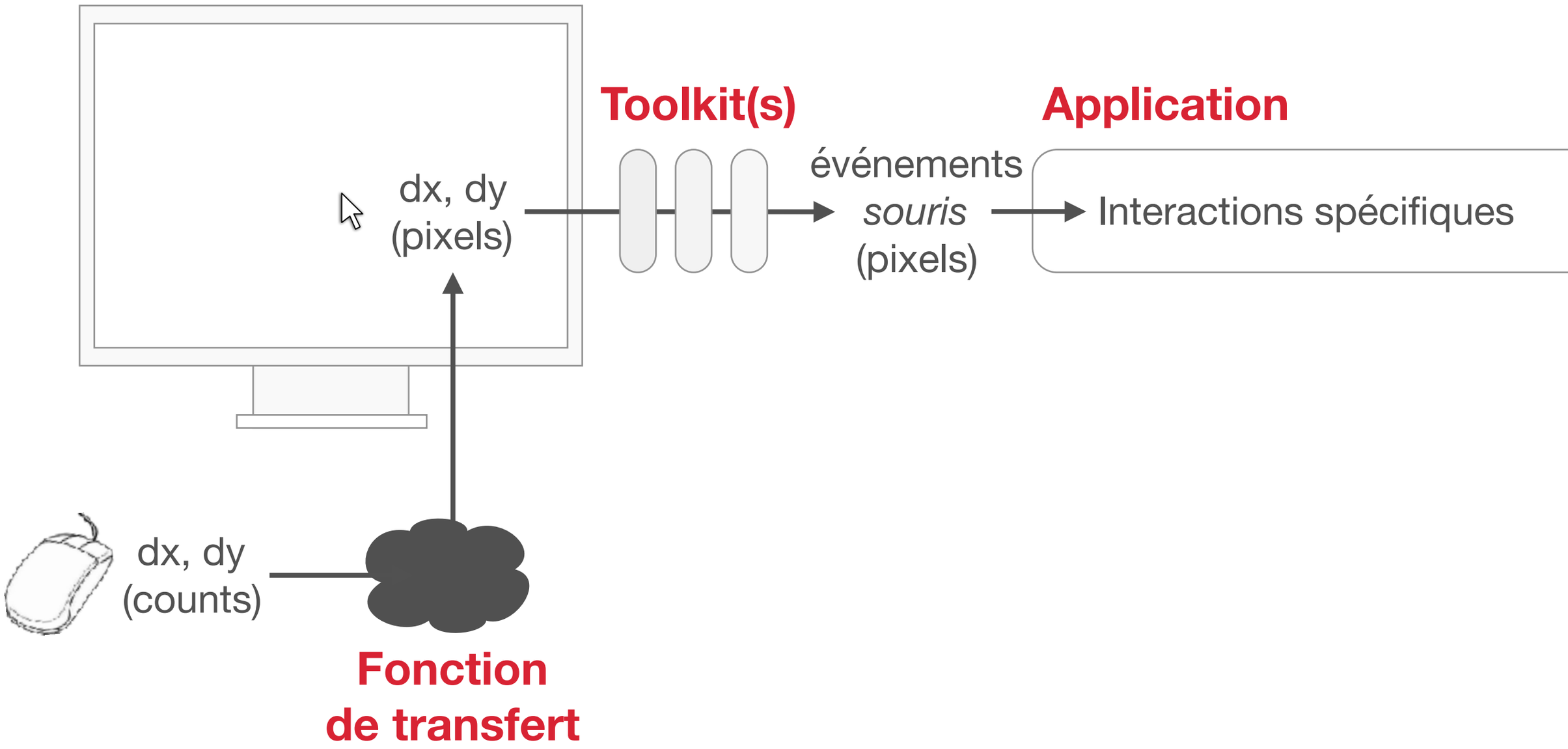


Les périphériques restent branchés à très bas niveau

On pourrait faire autrement : ICON (Dragicevic, 2001) permet de simuler des périphériques dont on ne dispose pas ou de reconfigurer dynamiquement les applications pour en utiliser de nouveaux



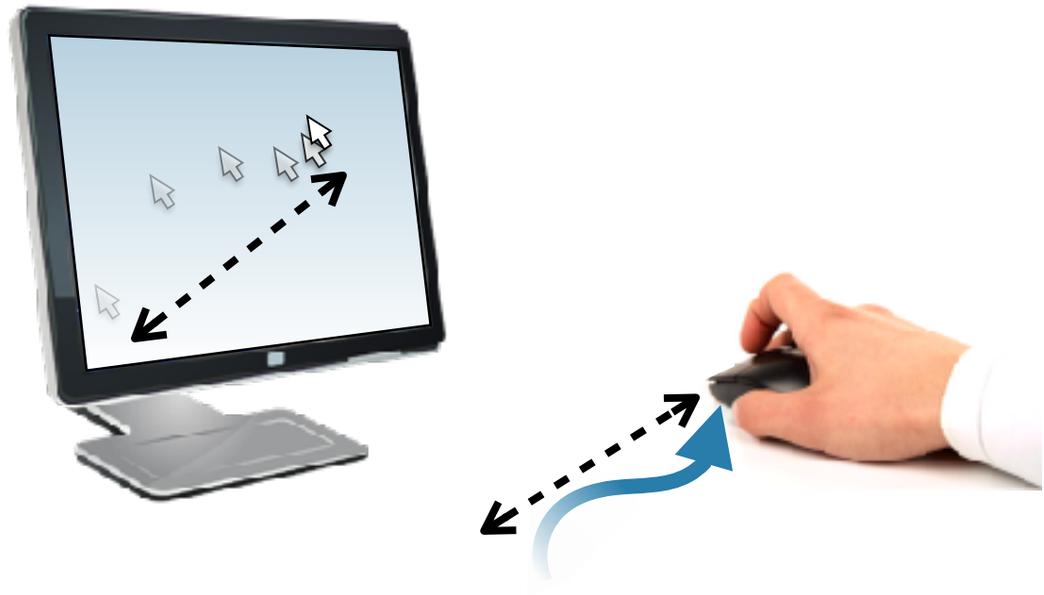
Pointage indirect : du travail en perspective



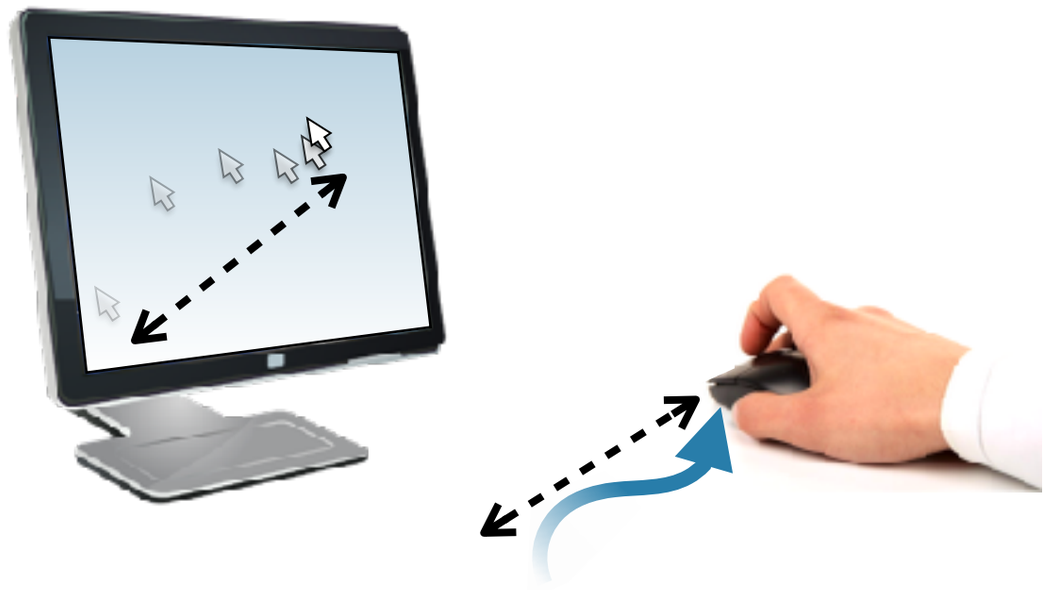
Fonctions de transfert pour le pointage indirect

Gain Contrôle-Affichage

- $V_{curseur} = V_{entrée} \times gain$



Transfer functions for cursor control

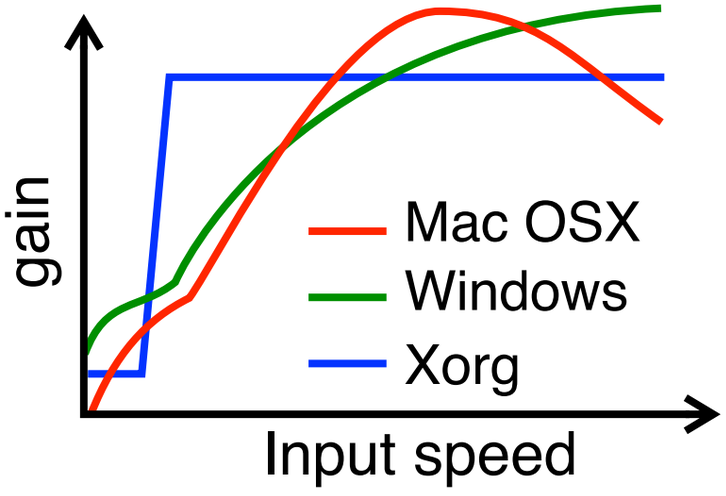


Gain Contrôle-Affichage

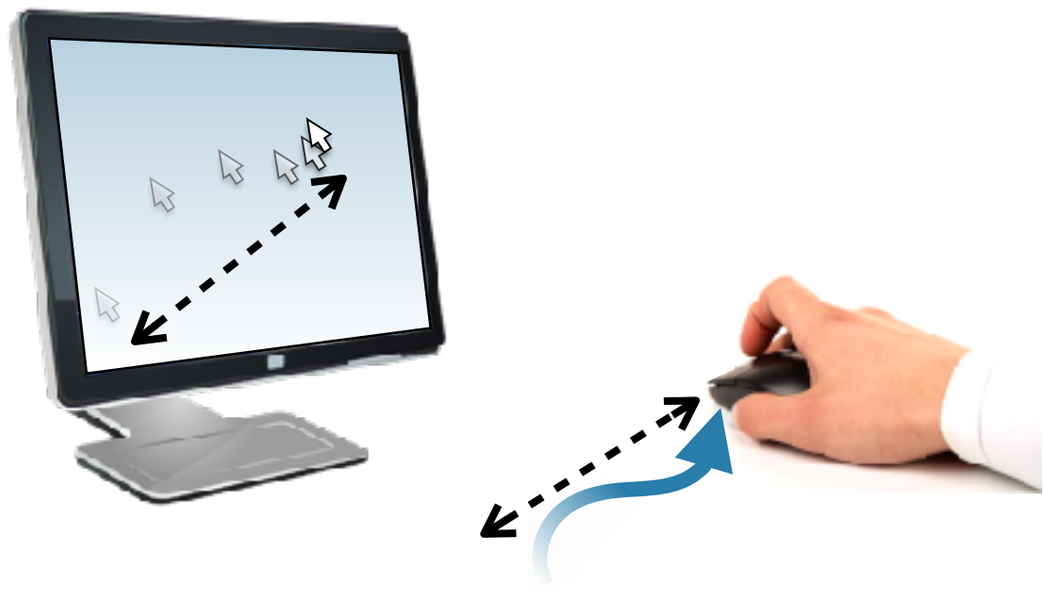
- $V_{curseur} = V_{entrée} \times gain$

Acceleration de curseur

- $V_{curseur} = V_{entrée} \times \mathbf{gain(V_{entrée})}$
- Plusieurs fonctions dans les OSs
- Complexe, modifications itératives
- Discret, mais essentiel



Fonctions de transfert pour le pointage indirect

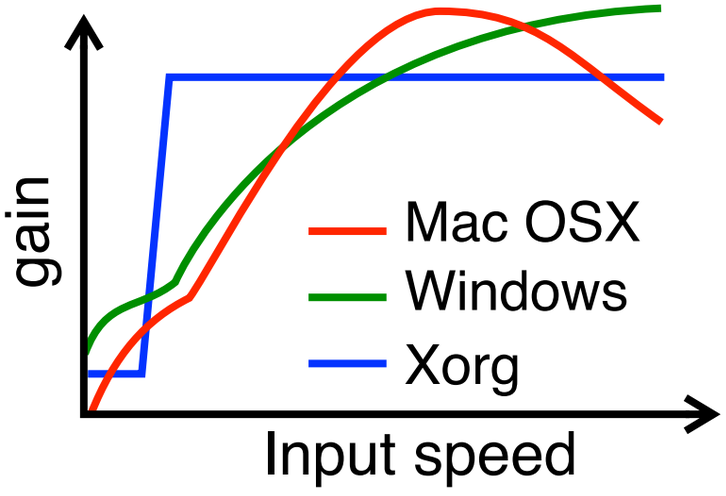


Gain Contrôle-Affichage

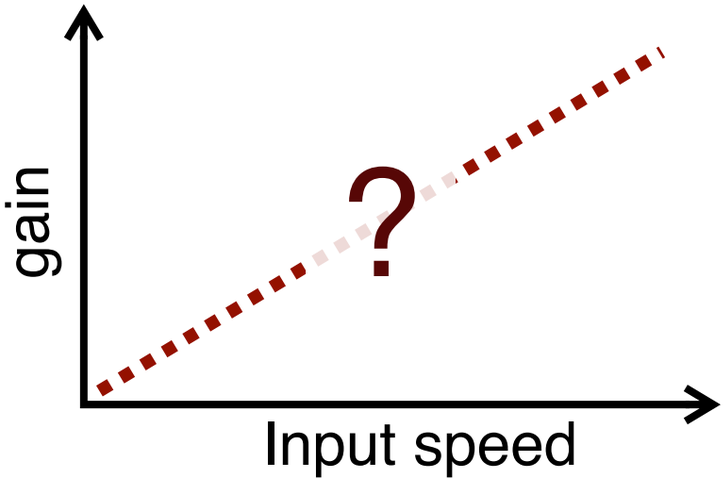
- $V_{curseur} = V_{entrée} \times gain$

Acceleration de curseur

- $V_{curseur} = V_{entrée} \times \mathbf{gain(V_{entrée})}$
- Plusieurs fonctions dans les OSs
- Complexe, modifications iteratives
- Discret, mais essentiel

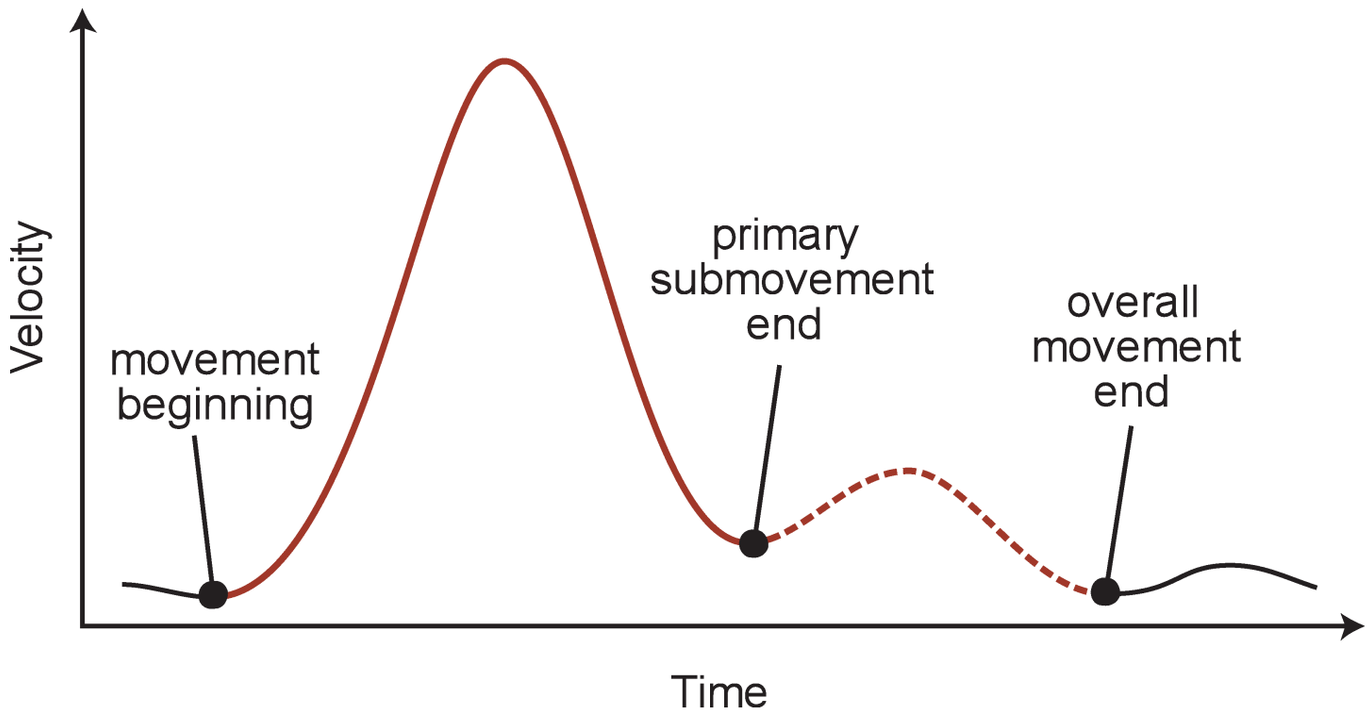


Aucune ligne de conduite admise



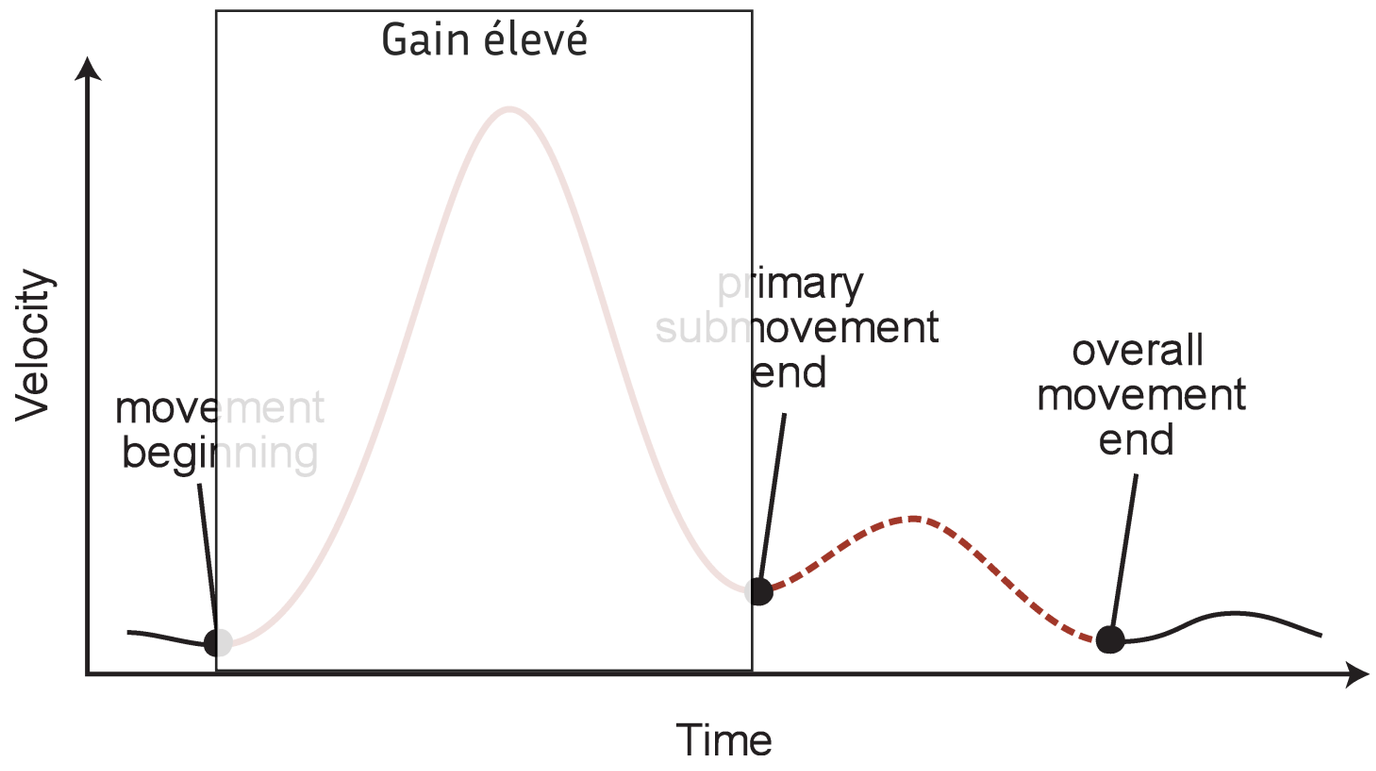
Fonctions de transfert pour le pointage indirect

Les fonctions actuelles ont été a priori conçues sur un principe similaire au *modèle d'impulsion initiale optimisée* de Meyer et al (1988)



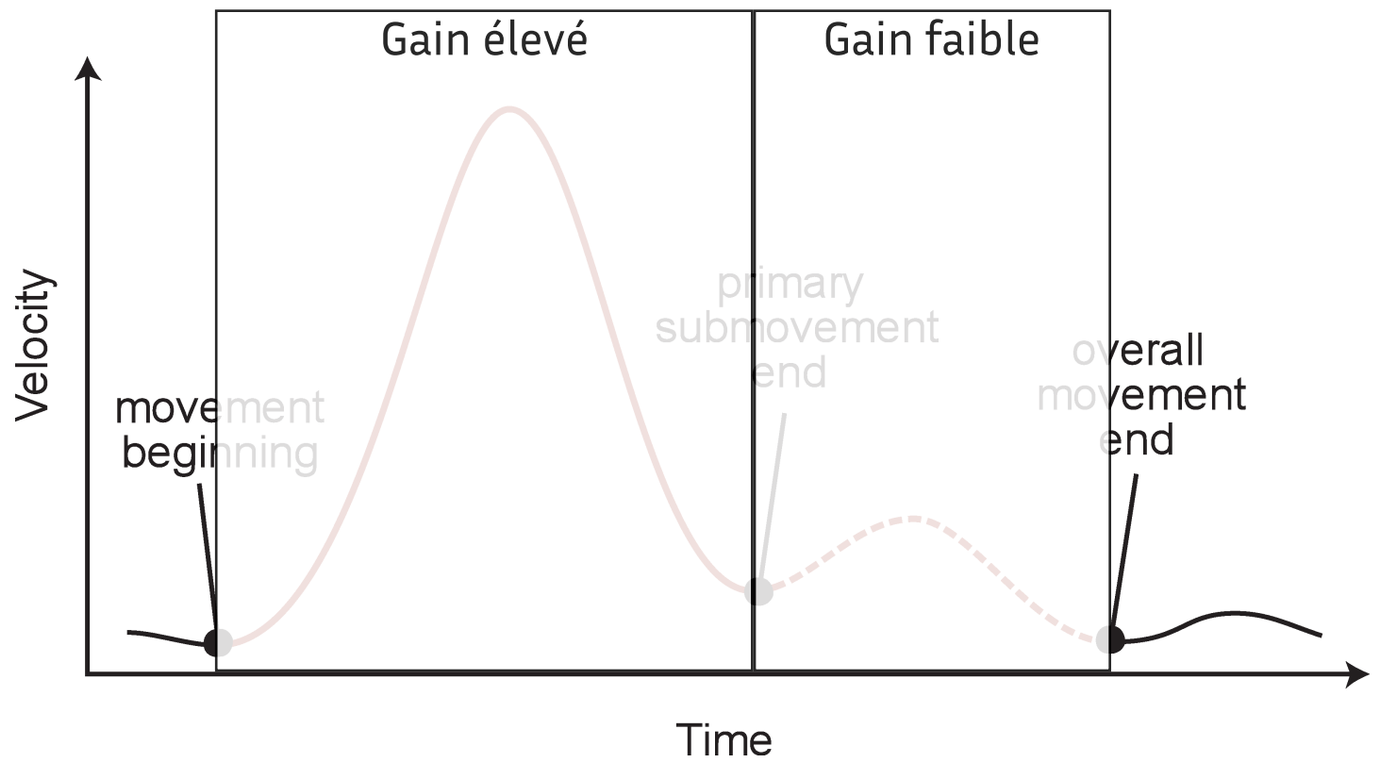
Fonctions de transfert pour le pointage indirect

Les fonctions actuelles ont été a priori conçues sur un principe similaire au *modèle d'impulsion initiale optimisée* de Meyer et al (1988)



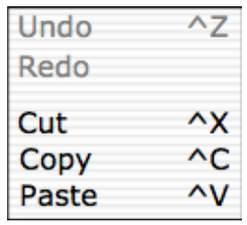
Fonctions de transfert pour le pointage indirect

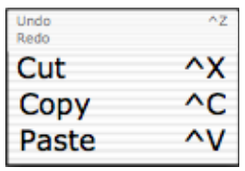
Les fonctions actuelles ont été a priori conçues sur un principe similaire au *modèle d'impulsion initiale optimisée* de Meyer et al (1988)



Elles pourraient être conçues autrement

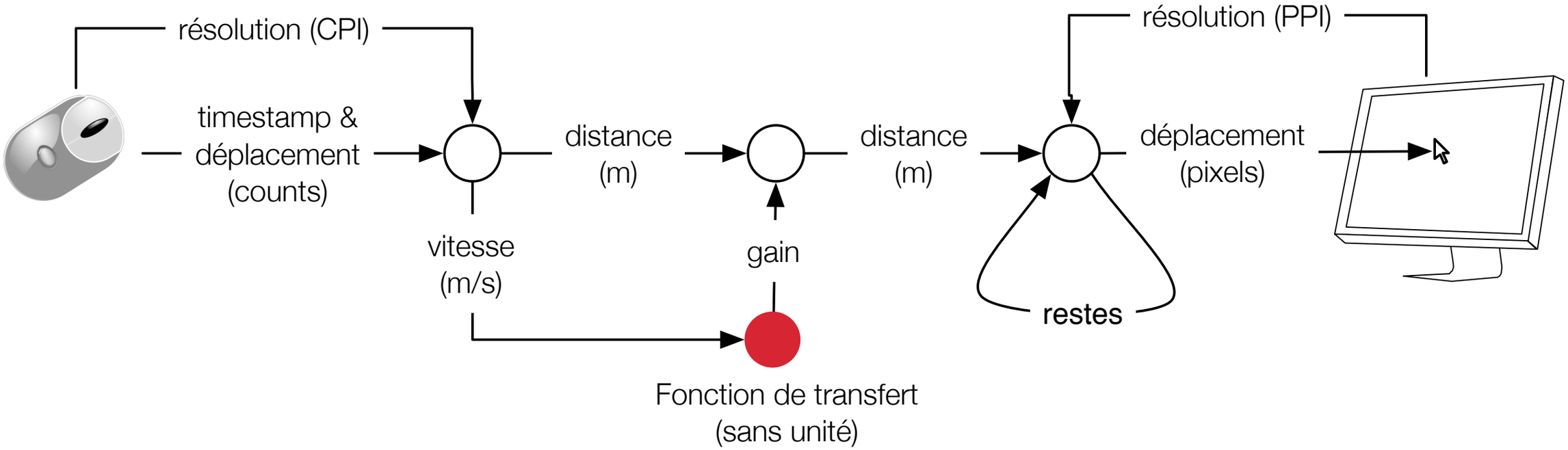
Principe du *pointage sémantique* (Blanch et al., 2004) : adapter le gain en fonction des objets à proximité du pointeur





Autre possibilité : avec le pointage vectoriel, Guiard et al., (2004) proposent de sauter les espaces vides

En théorie, donc…



En pratique ?

Pointer Ballistics for Windows XP

Updated: October 31, 2002

Archived paper. No warranty is made as to technical accuracy of content or currency of URLs.

On This Page

- Summary of Ballistic Performance
- Basic Theory of Pointer Ballistics for Windows XP
- Precision M
- Summary of
- Resources

```
bool SetupAcceleration (OSData * data, IO
    const UInt16 *      lowTable = 0;
    const UInt16 *      highTable;

    SInt32      x1, y1, x2, y2, x3, y3;
    SInt32      prevX1, prevY1;
    SInt32      upperX, upperY;
    SInt32      lowerX, lowerY;
    SInt32      lowAccl = 0, lowPoints =
    SInt32      highAccl, highPoints;
    SInt32      scale;
    UInt32      count;
    Boolean     lower;

    SInt32      devScale, crsrScale;
    SInt32      scaledX1, scaledY1;
    SInt32      scaledX2, scaledY2;

    CursorDeviceSegment *       segments
    CursorDeviceSegment *       segment;
    SInt32                      segCount

    if( !data || !resolution)
        return false;
```

AccelerationProfile [integer]

device property: Device Accel Profile

Select the acceleration profile by number. Default is 0, except if

In this section, *threshold* and *acceleration* specify the correspon

0. **classic** (the default) similar to old behaviour, but more predict

-1 **none**

no velocity-dependent pointer acceleration or deceleration

1. **device-dependent**

available if the hardware driver installs it. May be coming

2. **polynomial**

Scales polynomial: velocity serves as the coefficient, acce

3. **smooth linear**

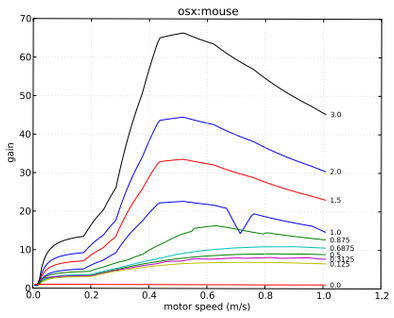
scales mostly linear, but with a smooth (non-linear) start.

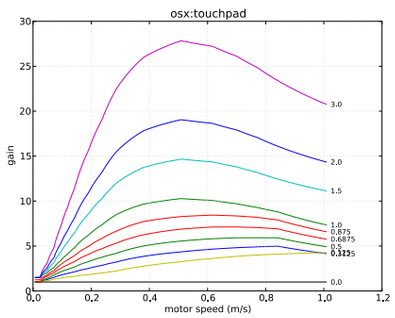
EchoMouse et Libpointing (Casiez & Roussel, 2011)

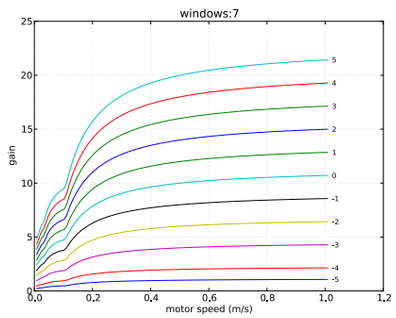
EchoMouse : une souris USB à laquelle on peut dire ce qu'elle doit envoyer au système

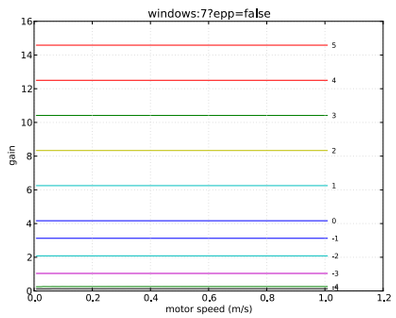
Libpointing : une librairie OS X, Linux et Windows pour

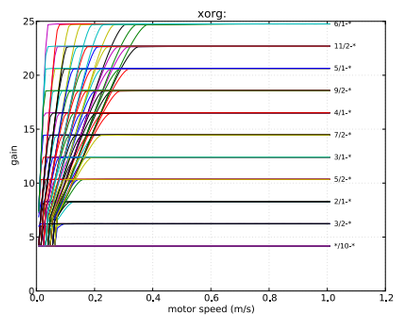
- accéder directement aux périphériques de pointage
- obtenir des informations de configuration de ces périphériques et des écrans (e.g. résolution)
- répliquer et comparer les fonctions de transfert existantes





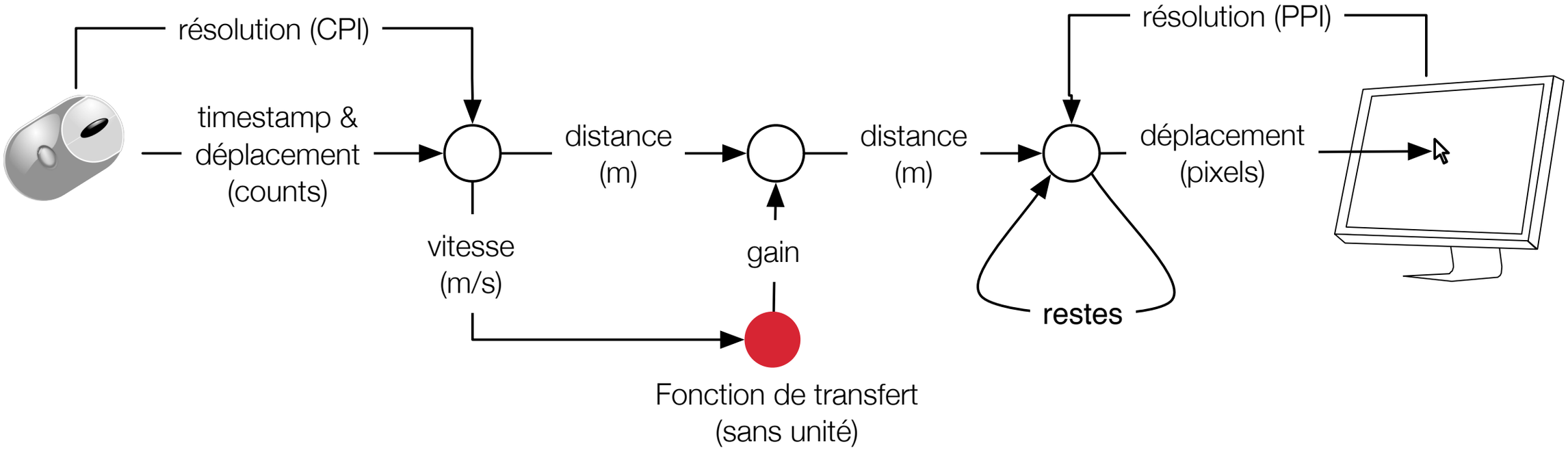




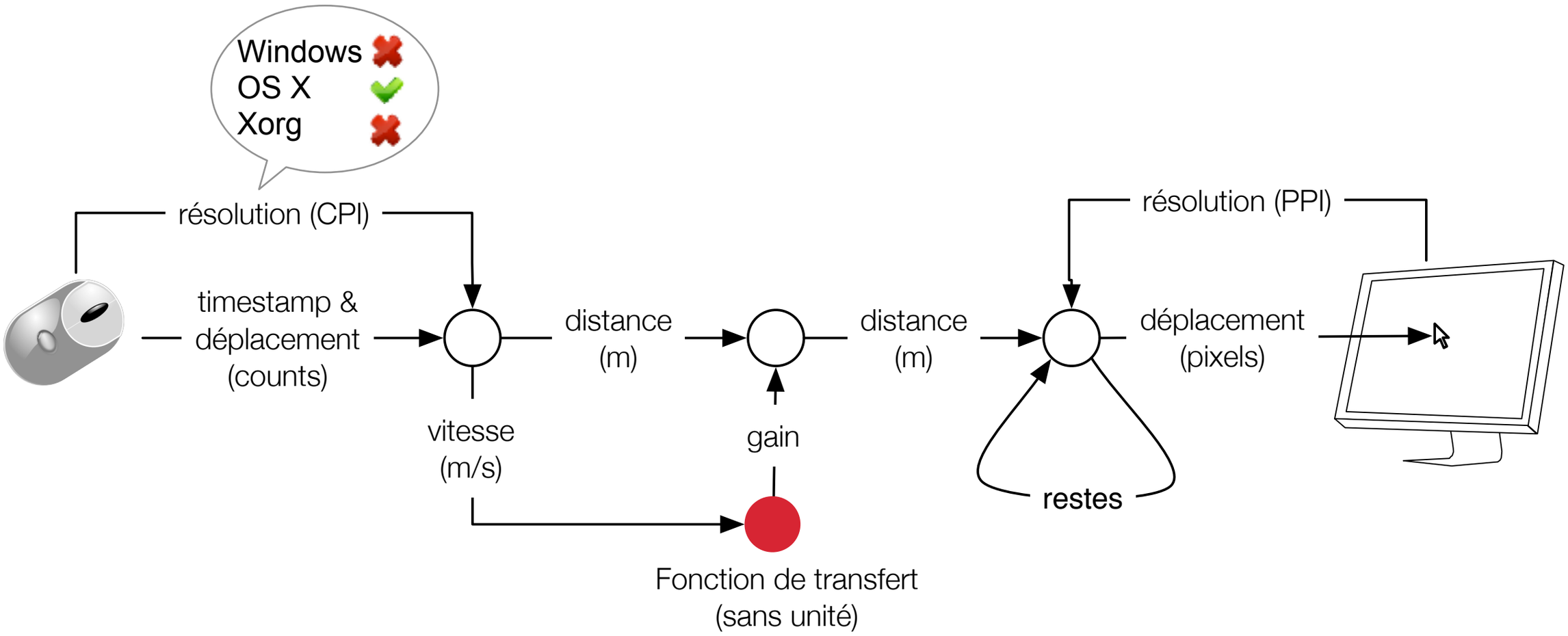


http://libpointing.org/

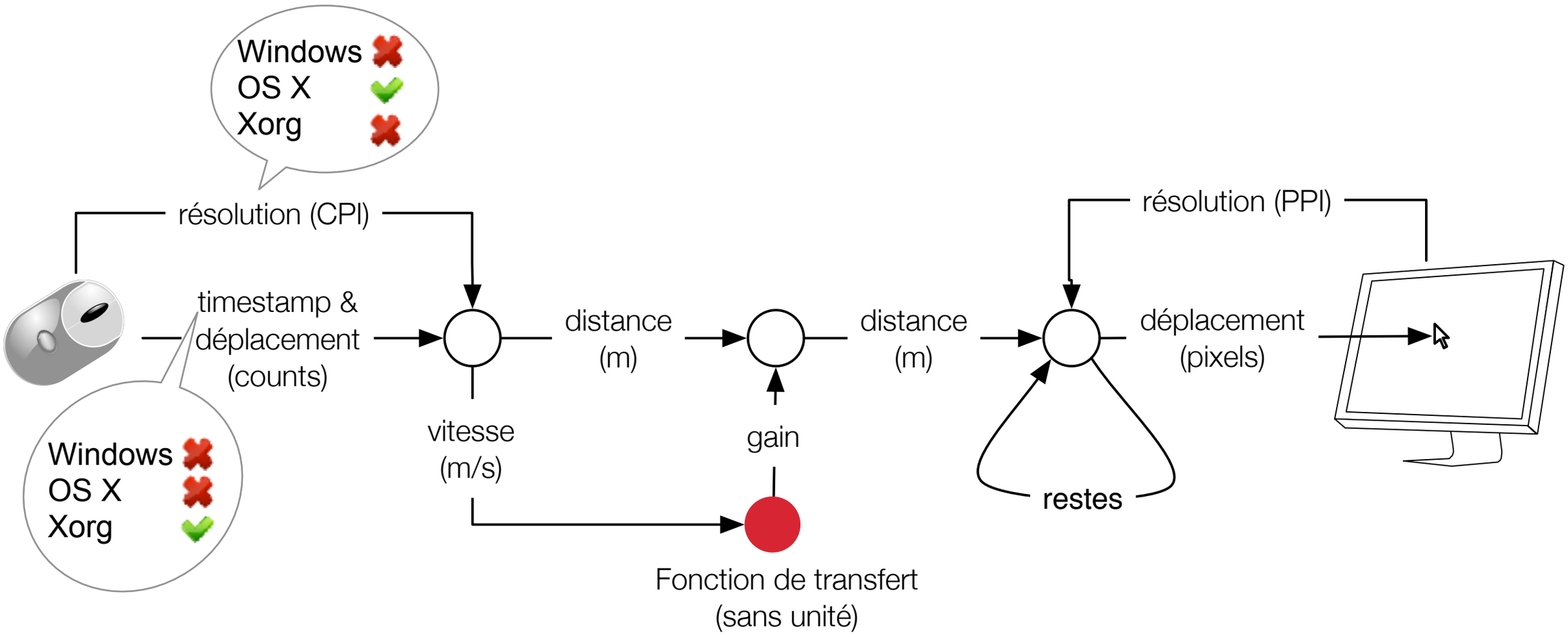
En pratique


résolution (CPI)
timestamp &
déplacement
(counts)
distance
(m)
vitesse
(m/s)
gain
distance
(m)
Fonction de transfert
(sans unité)
résolution (PPI)
déplacement
(pixels)
restes

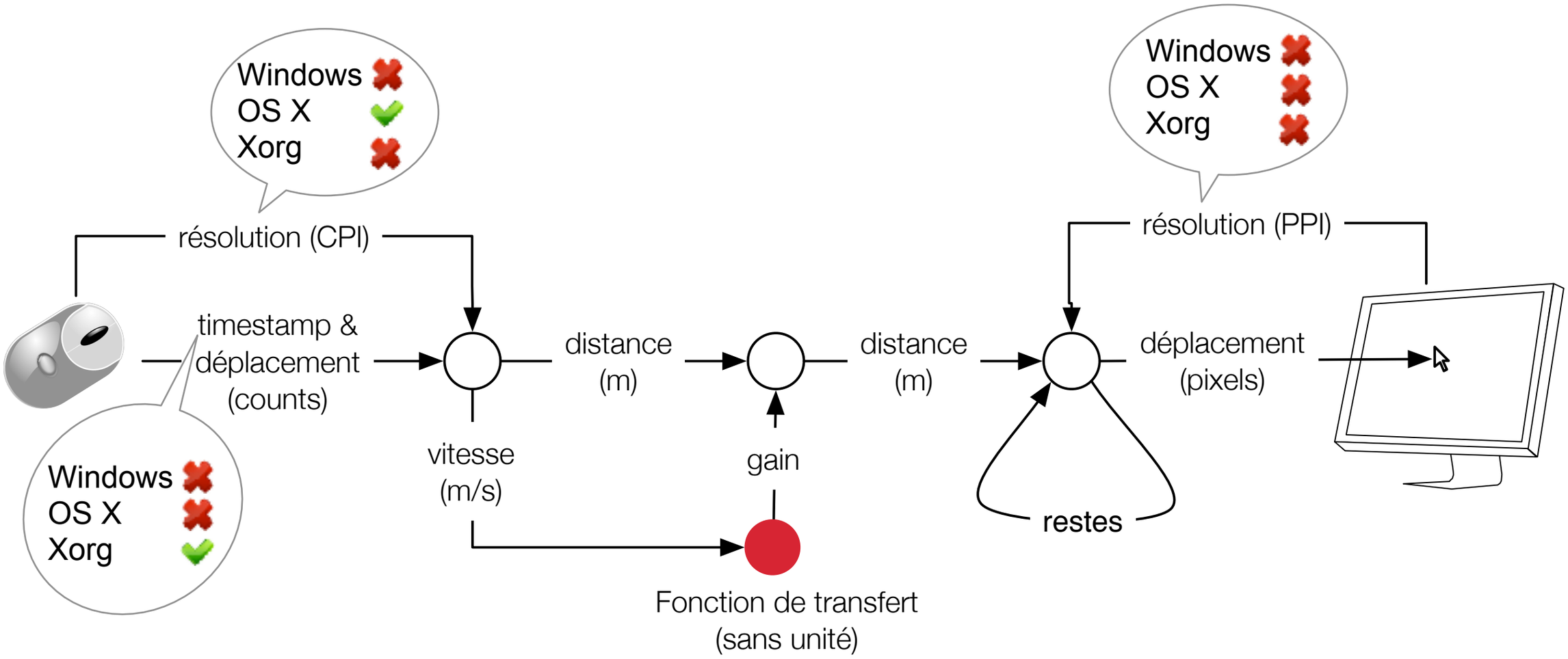
En pratique


Windows
OS X
Xorg
résolution (CPI)
timestamp &
déplacement
(counts)
vitesse
(m/s)
distance
(m)
gain
distance
(m)
résolution (PPI)
déplacement
(pixels)
restes
Fonction de transfert
(sans unité)

En pratique


Windows
OS X
Xorg
résolution (CPI)
timestamp &
déplacement
(counts)
Windows
OS X
Xorg
vitesse
(m/s)
distance
(m)
gain
Fonction de transfert
(sans unité)
distance
(m)
résolution (PPI)
déplacement
(pixels)
restes

En pratique


Windows
OS X
Xorg
résolution (CPI)
timestamp &
déplacement
(counts)
Windows
OS X
Xorg
distance
(m)
vitesse
(m/s)
gain
Fonction de transfert
(sans unité)
distance
(m)
Windows
OS X
Xorg
résolution (PPI)
déplacement
(pixels)
restes

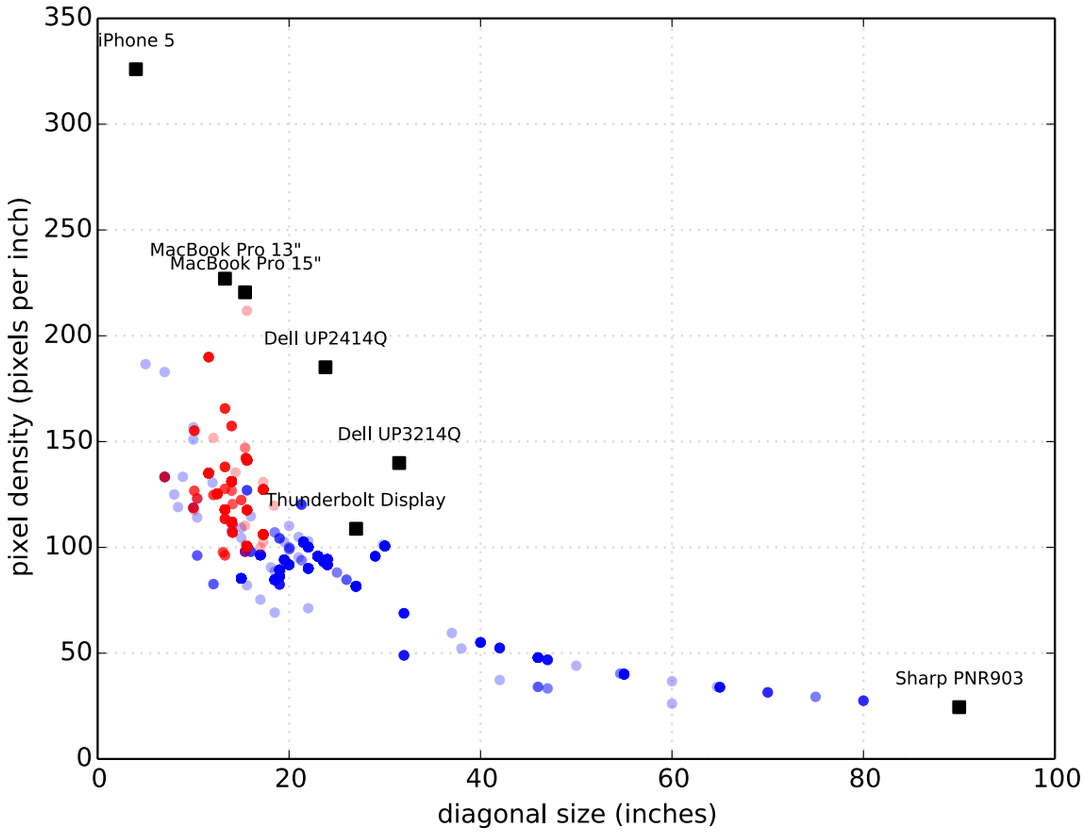
Bilan sur le pointage indirect

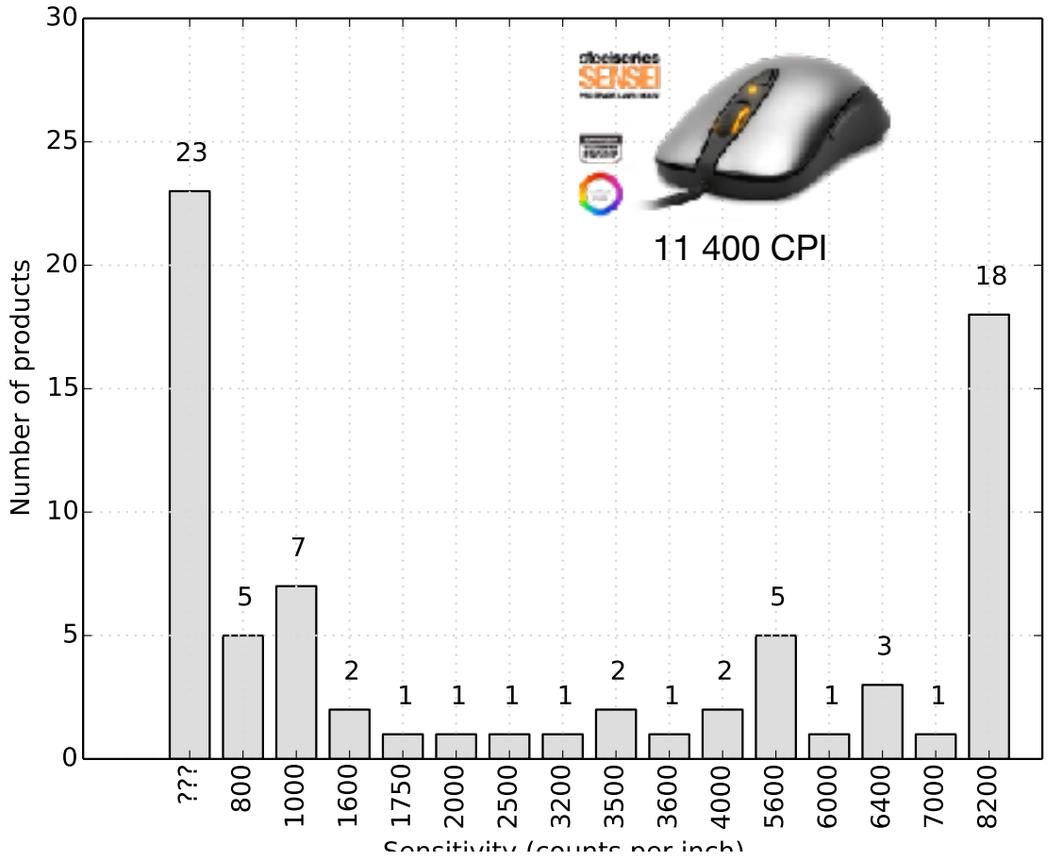
Les fonctions de transfert sont actuellement le seul mécanisme de facilitation du pointage disponible sur toutes les plateformes

Il y a des différences significatives entre les fonctions disponibles

Pourquoi la résolution des périphériques n’est-elle pas utilisée ?

Actuellement, sur le marché

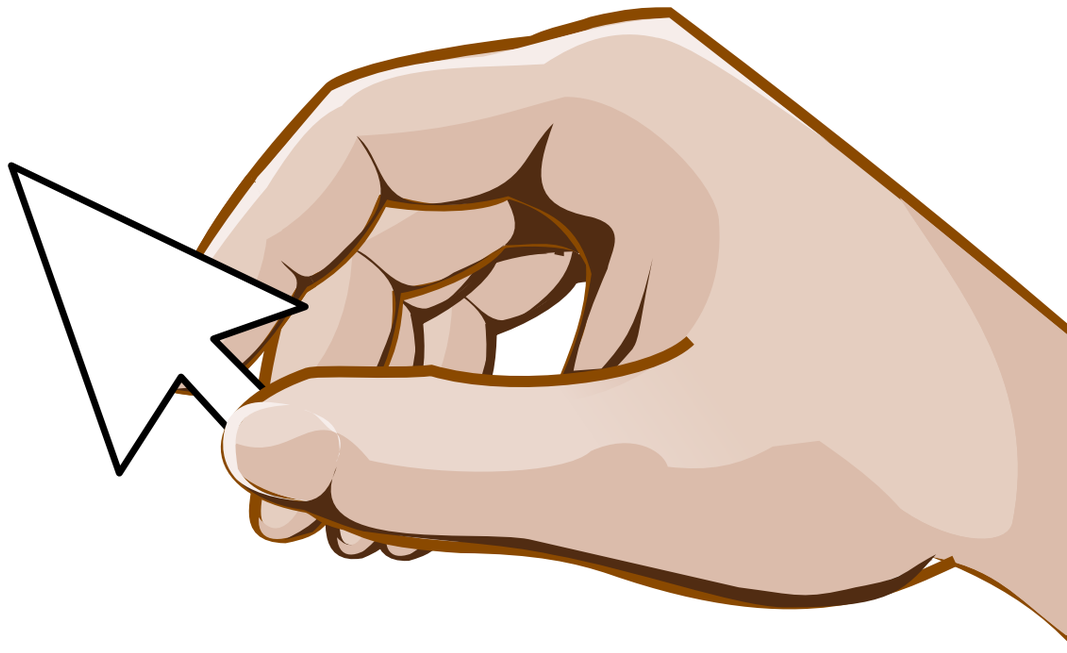




Il y a du potentiel, mais il n’est pas exploité

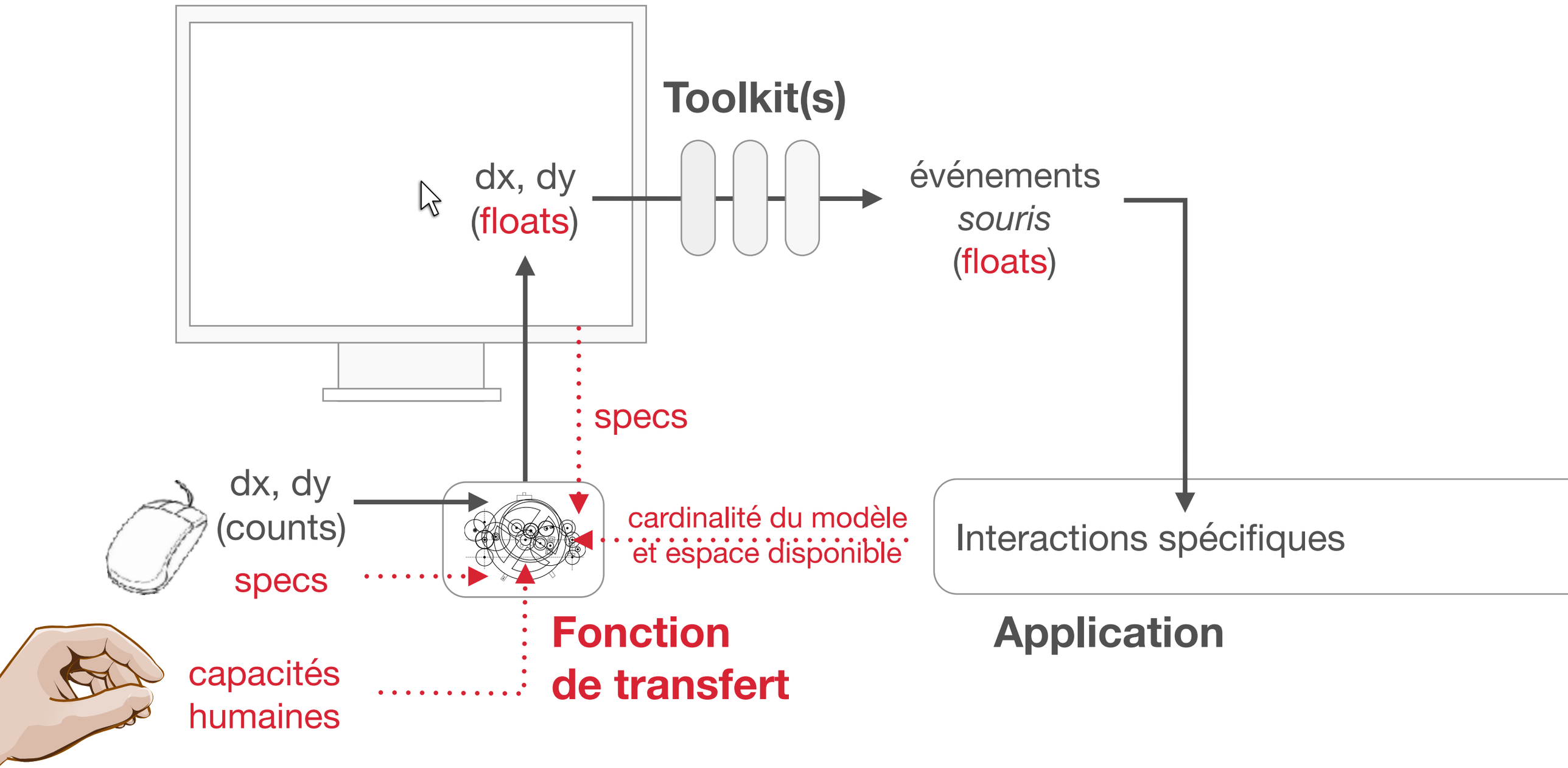
Le problème, c’est qu’on interagit à travers un pointeur

Le dispositif de pointage permet d’interagir avec le pointeur à l’écran, lequel permet d’interagir avec les données représentées à l’aide de pixels



On pourrait faire de l'interaction *subpixel*

(Roussel et al., 2012 ; Aceituno et al., 2013)



Gestion de fenêtres : que d'occasions manquées…

Rooms (ToG 1986)

Elastic windows (CHI 1997)

Dynamic space management (UIST 2000)

The task gallery (CHI 2000)

Tabbed, rotated, peeled-back, snapped and zipped windows (UIST 2001)

VideoWorkspace/Ametista (IHM 2002, CLIHC 2003)

Wincuts (CHI 2004)

Fold-and-drop (UIST 2004)

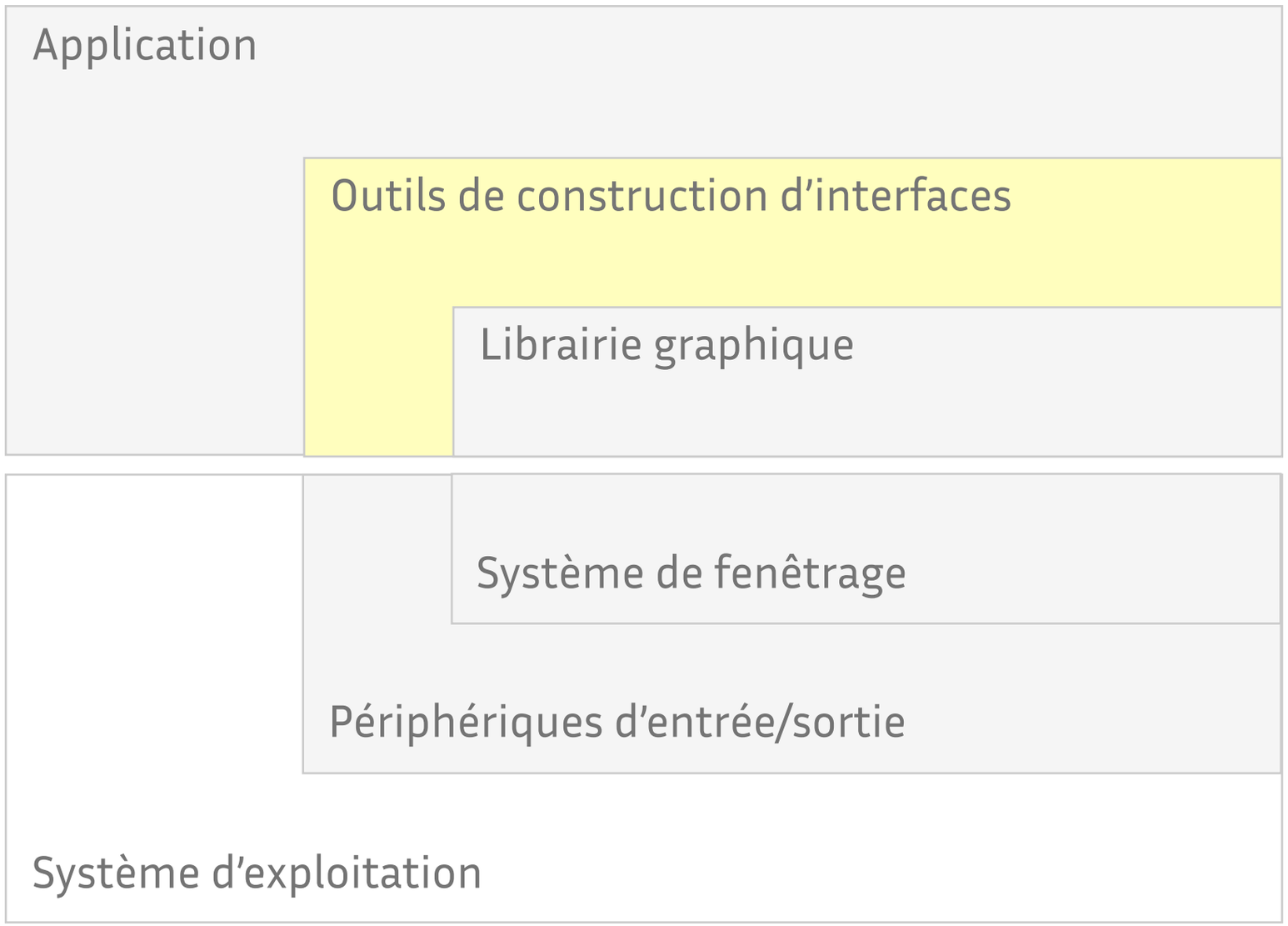
Scalable fabric (AVI 2004)

Shrinking window operations (AVI 2004)

Metisse (2005)

Mudibo (CHI 2005)

BumpTop (CHI 2006)


Application
Outils de construction d’interfaces
Librairie graphique
Système de fenêtrage
Périphériques d’entrée/sortie
Système d’exploitation

Du périphérique de localisation à la manipulation directe

Combiné à un périphérique d'affichage, un périphérique de localisation permet à l'utilisateur de pointer, i.e. de désigner une zone de l'écran qui l'intéresse plus particulièrement

L'utilisation d'un effecteur (e.g. un bouton) permet de marquer explicitement son intérêt : on parle alors de *sélection*

En jouant sur le type d'effecteur utilisé, le déplacement du pointeur pendant son temps d'activation et les répétitions, il devient rapidement possible de spécifier des segments de droite, des zones rectangulaires, des trajectoires, etc.

Exemples

- click avec le bouton gauche de la souris
- click avec le bouton droit, ou avec le bouton gauche et la touche Shift du clavier
- press-drag-release
- double-click
- demi-click, click et demi, etc.

Du périphérique de localisation à la manipulation directe (suite)

La combinaison [écran graphique+clavier+souris] a permis l'émergence d'un style d'interaction : *la manipulation directe*

Principes fondamentaux (Shneiderman, 1983)

- représentation permanente des objets d'intérêt ;
- utilisation d'actions physiques (e.g. pointage et sélection à la souris) plutôt que de commandes à la syntaxe complexe ;
- opérations rapides, incrémentales et réversibles, dont les effets doivent être immédiatement visibles sur les objets ;
- apprentissage selon une approche progressive afin de permettre l'utilisation de l'interface même avec un minimum de connaissances.

Deux phrases qui résument assez bien la philosophie

- "Point and click instead of remember and type"
- "People are good at recognition, but tend to be poor at recall"

Lignes de commande vs. manipulation directe

Les interfaces à ligne de commande reposent sur une syntaxe verbe-sujet(-compléments), ce qui a quelques inconvénients

- introduit un mode (et donc un risque d'erreur en cas d'interruption)
- rend nécessaire une action de type Cancel (pour sortir du mode)
- impose deux changements du locus d'attention (en supposant que celui-ci était sur l'objet)

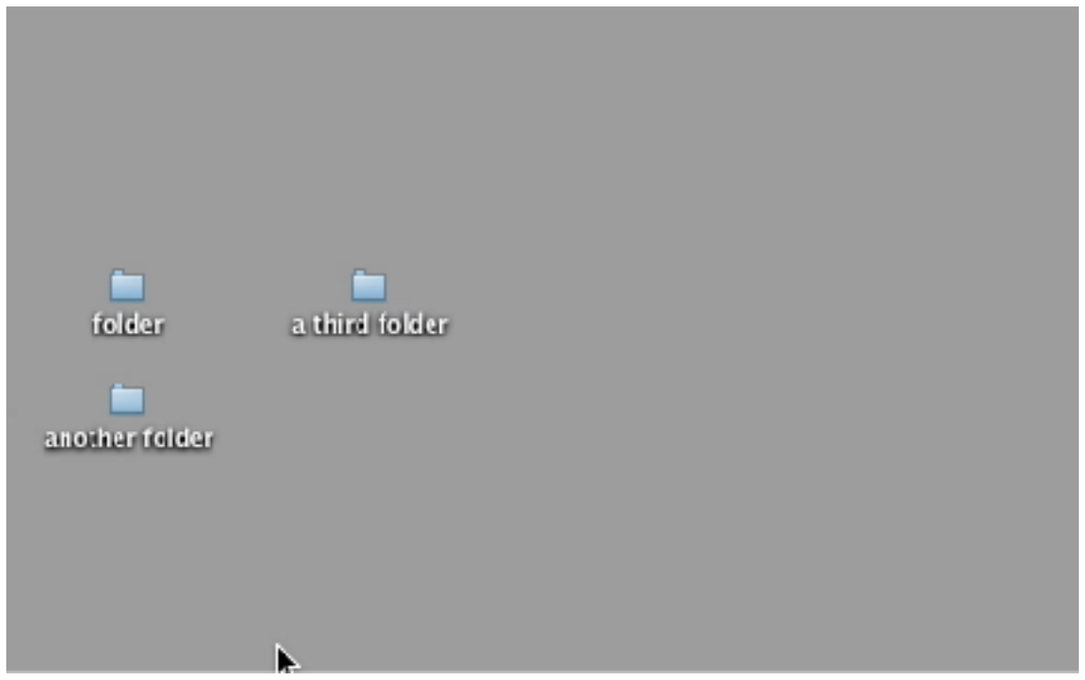
La manipulation directe repose sur une syntaxe sujet-verbe(-compléments) et des actions physiques

- pas de changement du locus d'attention, pas de mode
- des interactions génériques (e.g. jeter une icône à la corbeille), la commande effective dépendant de l'objet sélectionné
- les actions physiques rapides, incrémentales et réversibles facilitent un apprentissage rapide et/ou progressif par exploration ou démonstration

Importance du couplage perception/action

La continuité du couplage perception/action est un élément essentiel de la manipulation directe : les effets des opérations doivent être immédiatement visibles sur les objets

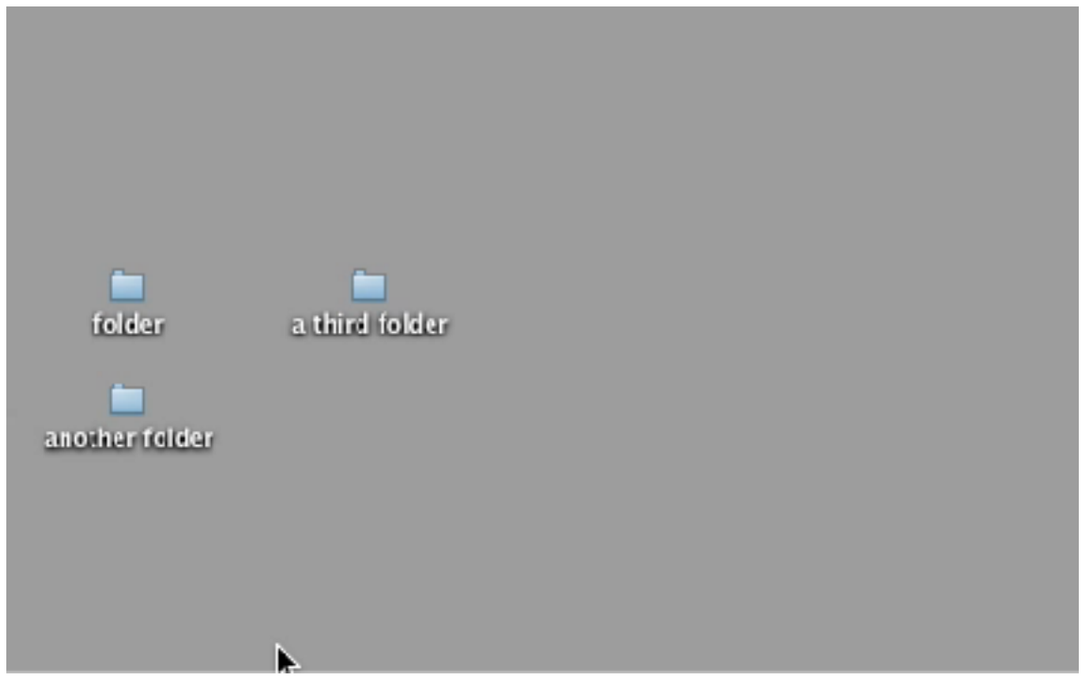
La mise en oeuvre de ce principe passe par des mécanismes de retour d'information (feedback mechanisms)



Importance du couplage perception/action

La continuité du couplage perception/action est un élément essentiel de la manipulation directe : les effets des opérations doivent être immédiatement visibles sur les objets

La mise en oeuvre de ce principe passe par des mécanismes de retour d'information (feedback mechanisms)



Modes d’interaction

Un mode est un état de l'interface dans lequel les actions de l'utilisateur sont interprétées de façon homogène et différentes des autres modes

Il existe différents types de modes

- modes spatiaux : la même action en différents endroits produit des effets différents
- modes temporels : la même action à des moments différents produit des effets différents
- micro-modes : modes liés à une action physique

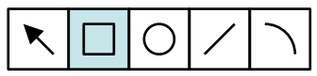
Une interface est une collection de modes

Modes d’interaction (suite)

Inconvénient des modes

- ils restreignent les fonctions disponibles à un moment donné
- ils obligent l'utilisateur a se souvenir des changements d'état de l'application pour connaître les effets de ses actions
- problèmes classiques : initiative et technique de changement de mode

Exemples

- touche Esc pour basculer entre les modes insert et command de l’éditeur vi
- palette d’outils 
- utilisation d’une pédale ou touche spéciale (Escape Meta Alt Control Shift)

Au-delà du pointage et de la sélection : les *widgets*

Un widget est un objet graphique interactif à trois facettes

- présentation : aspect graphique, plus ou moins paramétrable (e.g. Look and Feel Java)
- comportement : réactions aux actions de l'utilisateur, généralement, peu ou pas paramétrable
- interface d'application : lien avec l'application, notification des changements d'état

Objectifs : étendre le vocabulaire, faciliter l'apprentissage et la réutilisation de code

Exemple : les widgets de base du Macintosh (1984)

- bouton (*button*)
- menu déroulant (*pull-down menu*)
- case à cocher (*checkbox*), bouton radio (*radio button*)
- potentiomètre (*slider*), champ texte (*text field*)
- boîte de dialogue pour les fichiers (*file open/save dialog*)

Tâches élémentaires de l'interaction

Les widgets sont notamment utilisés pour réaliser de manière standard des tâches élémentaires d'interaction

A suivre : quelques exemples pour l'interaction graphique 2D

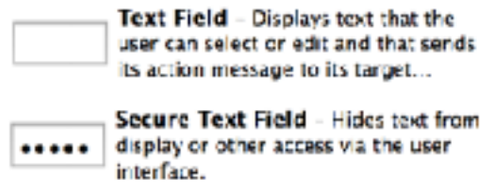
- saisie (textes, quantités, positions, tracés)
- sélection (par pointage, dans une liste, par saisie, par menu, exclusive, binaire)
- déclenchement (boutons, cliquer-tirer, entrée gestuelle)
- défilement (barres de défilement, défilement direct, défilement automatique)
- spécification d'arguments et de propriétés (boîtes d'alerte et d'information, boîtes de dialogue, boîtes de propriétés)
- transformations (poignées de manipulation)

Cette liste n'est pas exhaustive...

Tâches élémentaires : saisie

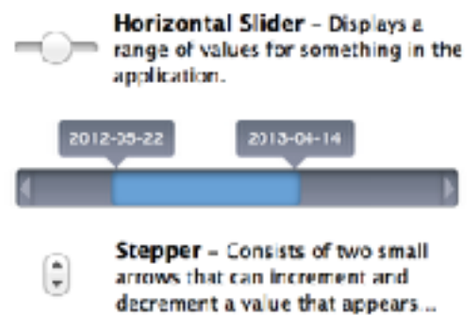
Saisie de texte : boîtes de saisie

- ligne simple ou multilignes
- formattage standard ou spécial (e.g. pour mot de passe)



Saisie de quantités

- potentiomètres (sliders, range sliders)
- steppers



Saisie de positions et de tracés : par échantillonnage du dispositif de pointage

La saisie peut être adaptée au type de valeur recherché (e.g. date, dimensions d'un tableau)

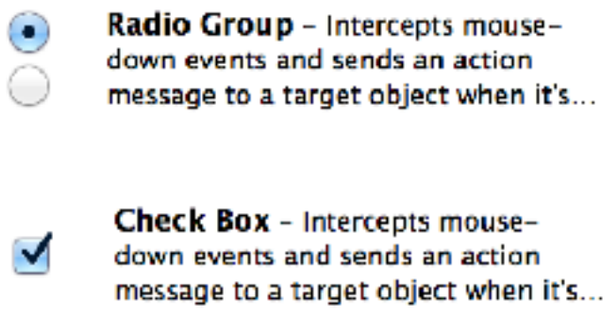
Tâches élémentaires : sélection

Choix d'un ou plusieurs éléments dans un ensemble

- cardinal fixe ou variable
- cardinal petit ou grand

Cardinal fixe

- boutons radio, cases à cocher
- menus

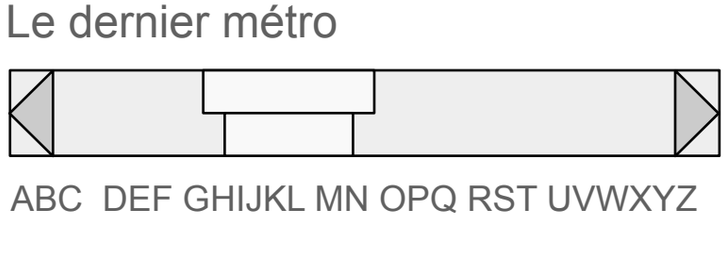


Cardinal variable

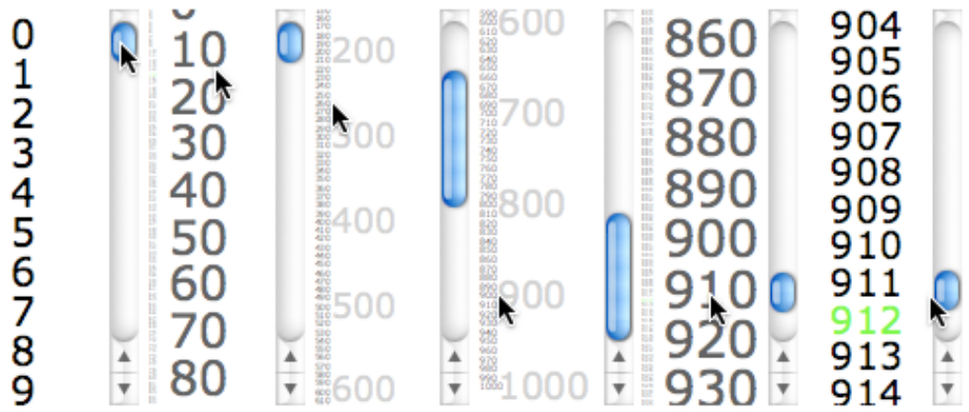
- pointage : avec feedback, éventuellement guidé/aidé, simple ou multiple (groupe ou intervalle) avec possibilité d'ajout ou de retrait
- listes, saisie ou combinaison des deux

Tâches élémentaires : navigation dans de grandes listes

L'alpha-slider (Ahlberg & Shneiderman, 1994) permet de sélectionner rapidement un élément dans une grande liste (e.g. de 13 à 24 secondes pour trouver un film dans une liste de 10.000 titres)



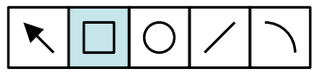
Autre exemple : OrthoZoom (Appert & Fekete, 2006) permet de naviguer facilement dans les 150000 lignes de texte des 37 pièces de Shakespeare



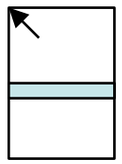
Tâches élémentaires : sélection dans des menus

Trois grands types de menus

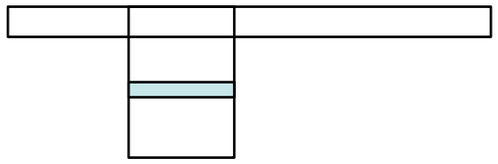
- fixes (palettes)



- surgissants (pop-up), généralement sous la souris



- déroulants (roll-down)

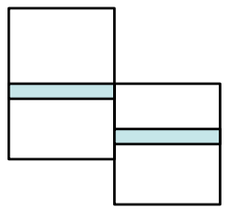


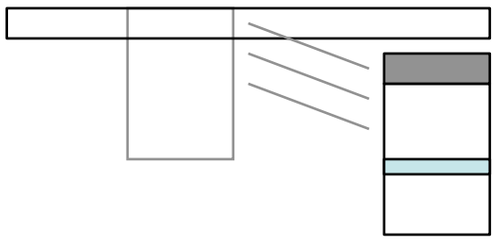
Les items sont généralement triés et groupés à l'aide de séparateurs

Des raccourcis clavier et mnémoniques (e.g. lettres soulignées) sont souvent disponibles

Les mécanismes de feedback sont là encore importants : items grisés, surlignés

Les menus pop-up et déroulants sont souvent hiérarchiques, parfois détachables

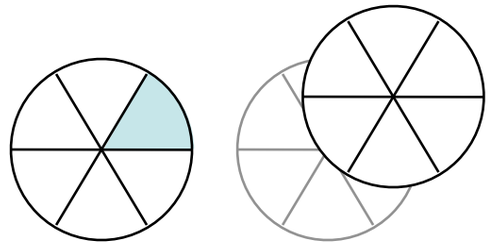




Tâches élémentaires : sélection dans des menus (suite)

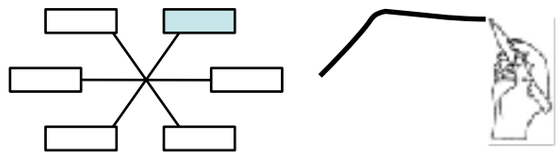
La plupart des menus sont organisés de manière linéaire, mais il existe toutefois quelques exceptions...

Pie menus (Hopkins, 1991)



- la sélection radiale est plus rapide que la sélection linéaire
- inconvénient : difficilement utilisable au-delà de 8 items par menu, difficile à positionner dans certains cas...

Marking menus (Kurtenbach, 1993)



- idée : permettre la sélection par geste sans affichage du menu
- transition aisée du mode novice au mode expert

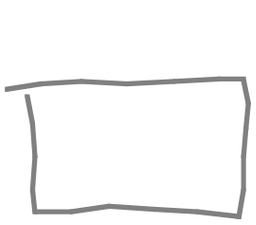
L'implémentation des menus circulaires pose de nombreux problèmes... mais il existe des solutions

Tâches élémentaires : déclenchement

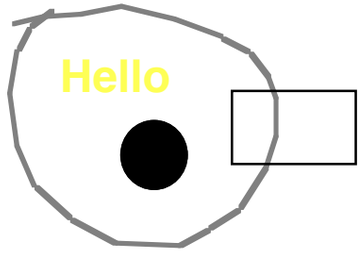
Boutons (et menus)

Glisser-déposer (drag-and-drop) : l'action dépend de la source et de la destination

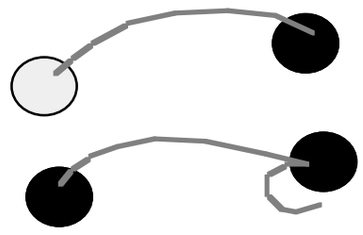
Entrée gestuelle : spécification simultanée de la commande et de l'objet



Création

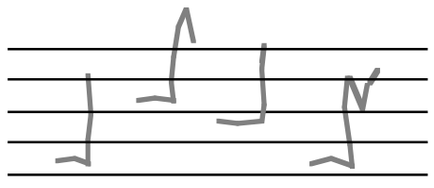


Sélection

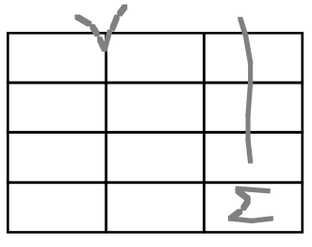


Déplacement, copie

Destruction



Editeur musical



Tableur

Tâches élémentaires : défilement

Barres de défilement

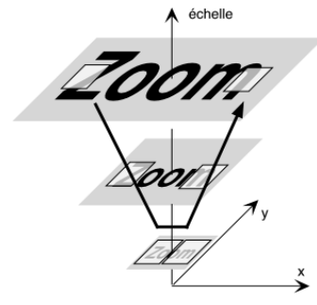
- sens du défilement ?
- découplage spatial

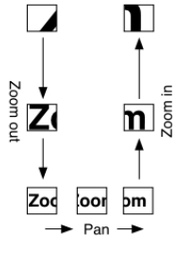
Défilement direct

- déplacement “à la main”
- défilement automatique

Zoom

- zoomer pour voir le détail, dézoomer pour voir le contexte
- différentes techniques de pan&zoom : SDAZ, OrthoZoom, control menus





Tâches élémentaires : défilement

Barres de défilement

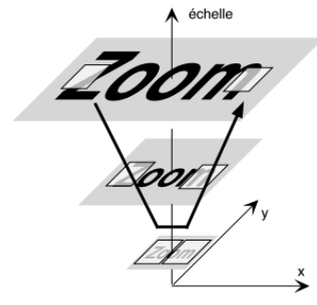
- sens du défilement ?
- découplage spatial

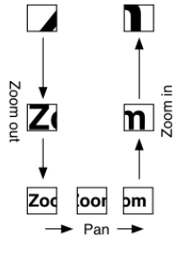
Défilement direct

- déplacement “à la main”
- défilement automatique

Zoom

- zoomer pour voir le détail, dézoomer pour voir le contexte
- différentes techniques de pan&zoom : SDAZ, OrthoZoom, control menus





Tâches élémentaires : défilement

Barres de défilement

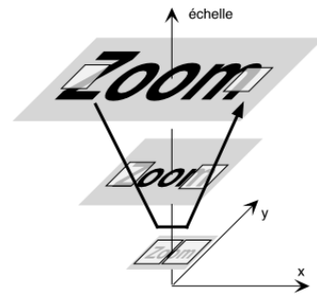
- sens du défilement ?
- découplage spatial

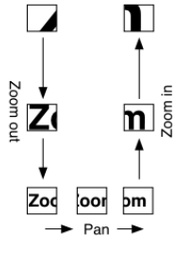
Défilement direct

- déplacement “à la main”
- défilement automatique

Zoom

- zoomer pour voir le détail, dézoomer pour voir le contexte
- différentes techniques de pan&zoom : SDAZ, OrthoZoom, control menus



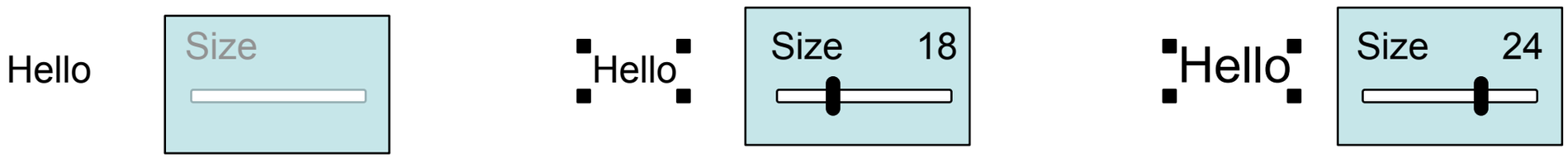


Tâches élémentaires : spécification d'arguments et propriétés

Boîtes de dialogue : champs + bouton(s) de type Ok, Apply, Cancel, etc.

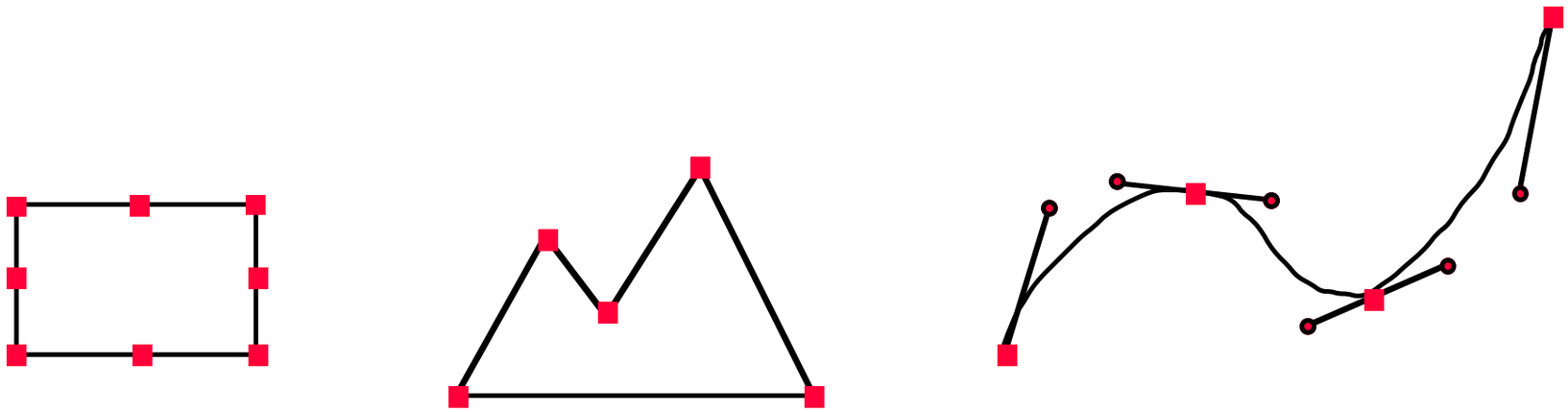
- boîtes modales ou non modales
- découplage temporel et spatial entre la spécification de la commande, de ses paramètres et de son exécution
- parties optionnelles, boîtes à onglets

Boîtes de propriétés :
effet immédiat des modifications sur les objets de la sélection



Tâches élémentaires : transformation

Poignées de manipulation et modificateurs permettant de changer la sémantique des actions effectuées (e.g. déplacer, changer la géométrie, dupliquer)



Au-delà des widgets : boîtes à outils de construction d'interfaces

Que trouve-t-on dans une boîte à outils ?

- des widgets réutilisables (e.g. boutons, menus)
- des services programmables (e.g. préférences, copier/coller)
- un modèle d'architecture permettant de structurer l'application
- un générateur et un ou plusieurs langages de scripts

Pourquoi utiliser une boîte à outils ? Pour optimiser l'un des points suivants

- temps de construction
- efficacité
- compacité
- robustesse
- correction
- confort d'utilisation
- versatilité

Boîtes à outils de construction d'interfaces

Il existe de nombreuses boîtes à outils

- Xt, Motif, GTK (C) sous Linux
- MFC (C++) et ses dérivés sous Windows
- Cocoa (Objective-C) sous OS X
- Tk (Tcl, Python, Perl et autres), Qt (C++) et Swing (Java) sur toutes les plate-formes

Exemples de services : placement des widgets, liens widgets - application, squelettes d'application, générateur de code

La plupart de ces boîtes à outils présentent deux limitations majeures

- la programmation est fastidieuse
- l'interaction se limite le plus souvent aux widgets pré-existants

Exemple d'interaction simple qui reste difficile à programmer : le drag-and-drop

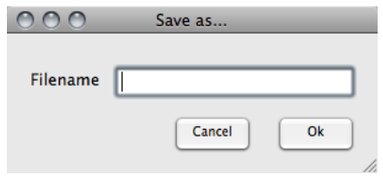
Exemple de service : placement des widgets

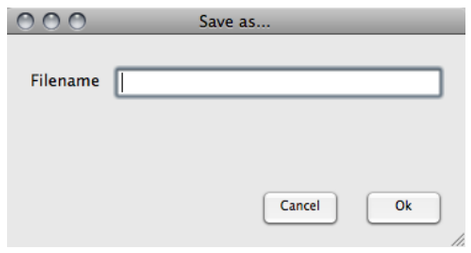
L'interface est généralement décrite par un arbre de widgets

- les noeuds sont des conteneurs (e.g. fenêtre, cadre)
- les feuilles sont des widgets simples (e.g. bouton, label, barre de défilement)

Gestionnaire de placement : placement géométrique des noeuds de l'arbre de widgets

- imbrication géométrique d'un widget fils dans son parent
- contrôle par le parent du placement de ses fils
- placement dynamique





Contraintes

- taille "naturelle" de chaque fils
- taille et position imposées par le parent
- autres contraintes spécifiées par le programmeur (e.g. grille, formulaire, ressorts)

Exemple de service : description/specification de l'interaction

Application non-interactive : "start, do something, stop"

Ancienne façon de voir les applications interactives : de temps en temps, le système pose des questions à l'utilisateur

Version moderne : l'utilisateur contrôle l'application, via l'interface

- l'application attend régulièrement un signe de l'utilisateur (notion de boucle principale dans laquelle sont traités les événements)
- le modèle de programmation événementielle peut être utilisé à l'intérieur même de l'application
- de nombreux états à conserver... c'est compliqué !

Comment décrire/spécifier l'interaction ?

Avec des machines à états

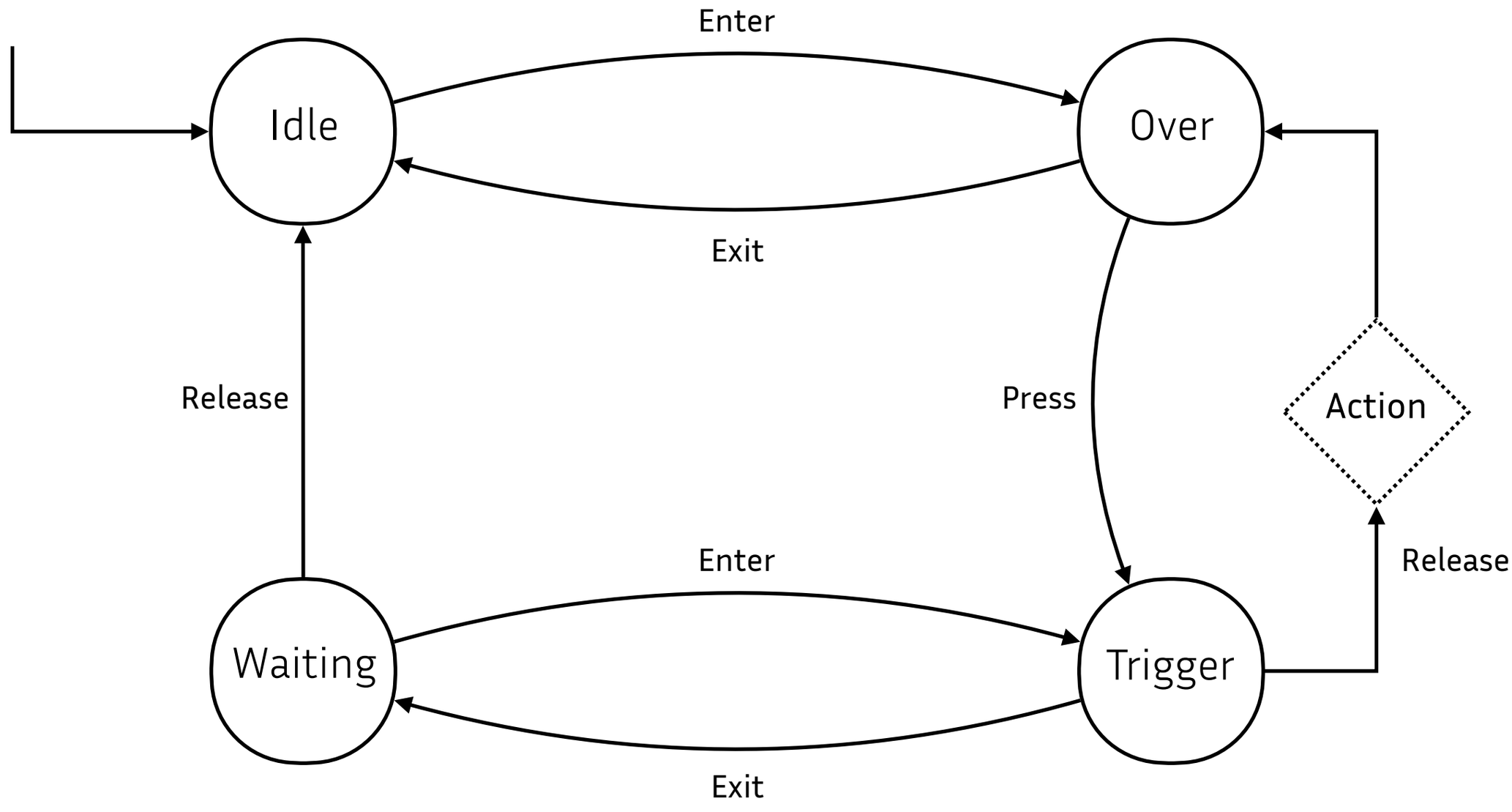
Une *machine à états* est un automate à états finis permettant de décrire formellement l'interaction avec un système

- chaque état de l'automate correspond à un état de l'interaction
- les transitions de l'automate sont déclenchées par des événements
- des conditions et actions peuvent être associées aux transitions

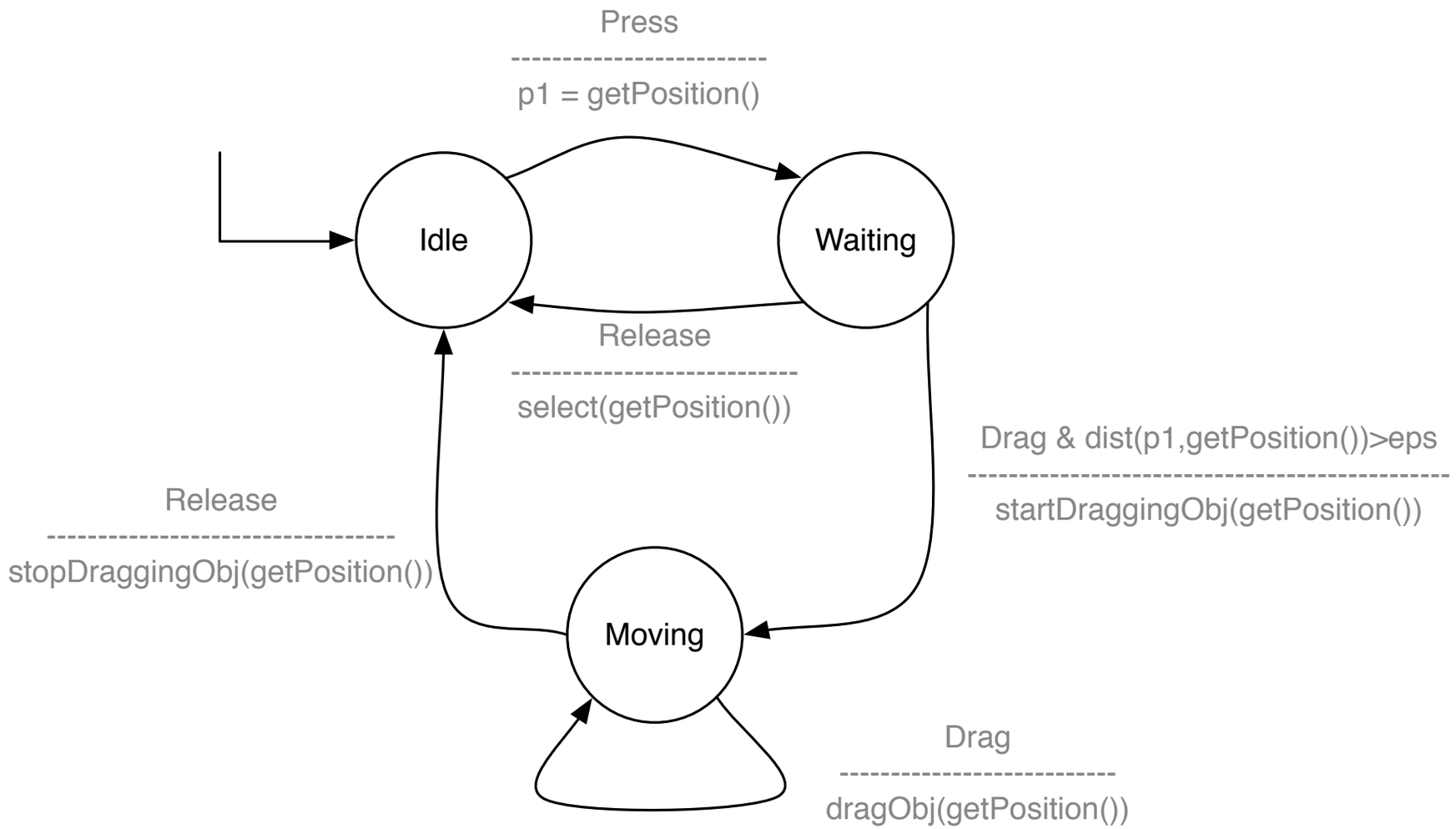
Intérêts

- possibilité de preuve et de validation formelle
- possibilité de lien direct avec l'implémentation

Exemple 1 : activation d'un bouton



Exemple 2 : sélection et déplacement d'un objet



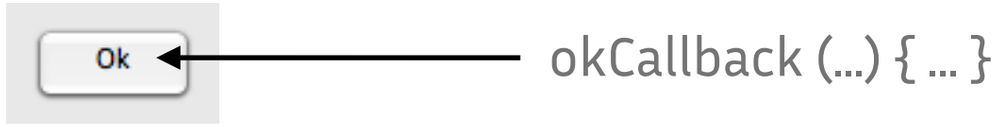
Difficultés liées aux machines à états

Les langages couramment utilisés ne permettent pas de décrire simplement les machines à états nécessaires aux interactions souhaitées

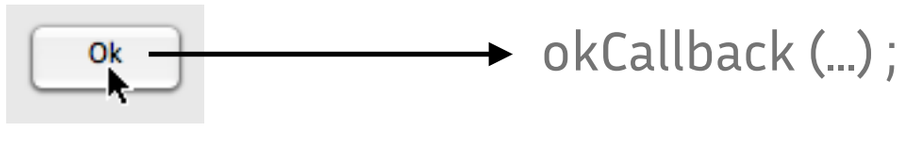
Celles-ci sont généralement le résultat implicite de liens entre les éléments d'interface et le reste de l'application établis par différents moyens

Liens widgets - application : fonctions de rappel

Les fonctions de rappel sont enregistrées dans le widget à sa création



Elles sont ensuite appelées lorsqu'une des opérations du widget est activée

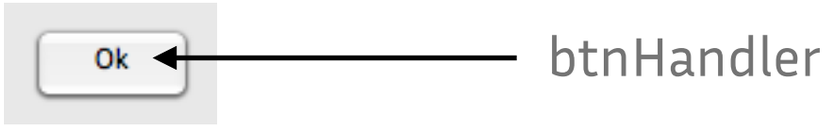


Inconvénients

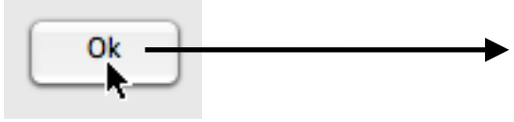
- un plat de spaghettis de callbacks
- la nécessité de passer par des variables globales pour partager des données entre fonctions de rappel

Liens widgets - application : envoi de messages

Un objet est associé au widget au moment de sa création



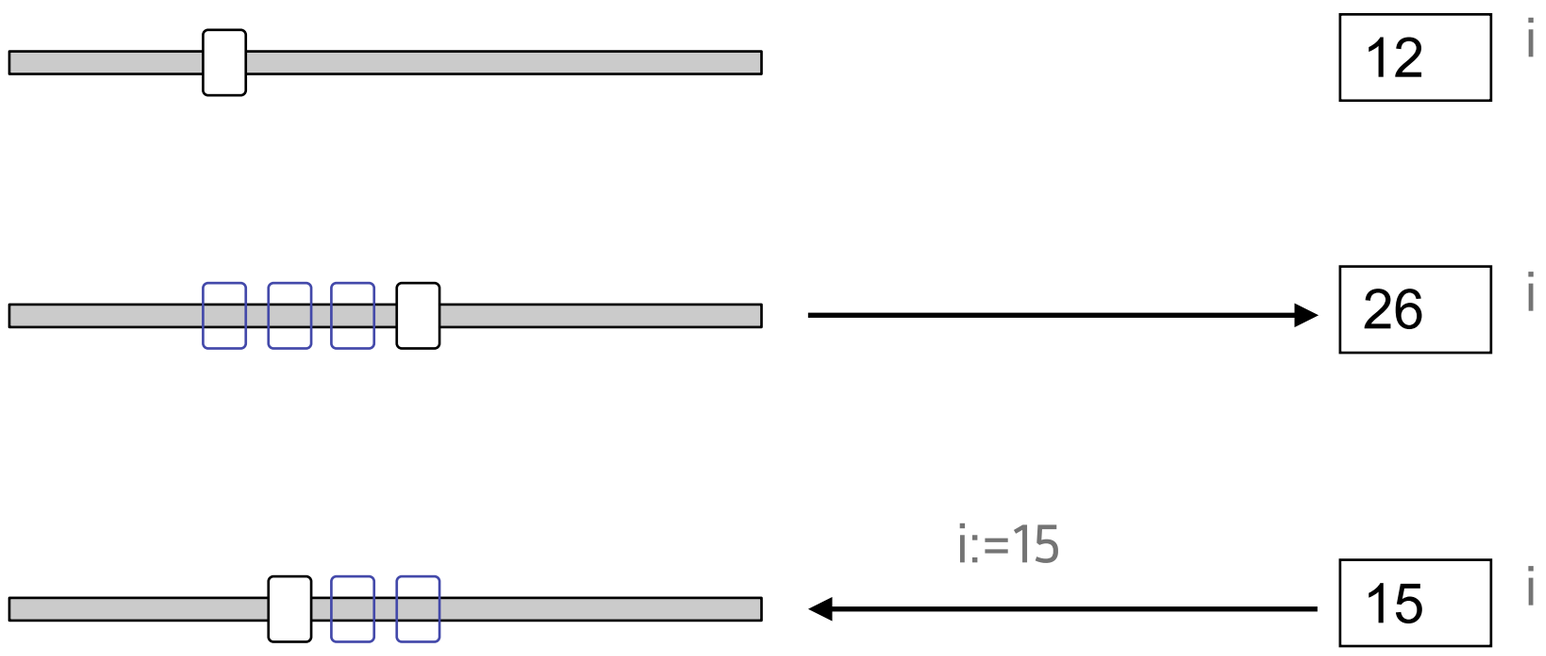
Des méthodes de cet objet seront appelées par le widget



Avantage : permet une bonne encapsulation des données

Liens widgets - application : valeurs actives

Une valeur active établit un lien bi-directionnel entre une variable d'état d'un ou plusieurs widget(s) et une variable du noyau fonctionnel



Avantage : vues multiples

Inconvénient : mécanisme rarement disponible et approche généralement limitée aux types simples (e.g. entiers)