Interaction graphique

Introduction

L'interaction graphique regroupe l'ensemble des moyens matériels et logiciels qui permettent d'interagir avec des représentations graphiques affichées sur un écran. Dans ce chapitre sont abordés : les dispositifs d'affichage et les périphériques d'entrée pour ce qui concerne le matériel ; la gestion des entrées au niveau logiciel et la notion de machine à états pour modéliser le traitement des entrées ; enfin une taxonomie détaillée des tâches d'interaction de la manipulation directe. La gestion de l'affichage lui-même ne sera abordée que très rapidement lors de l'étude des systèmes de fenêtrage : on pourra se reporter à un cours d'infographie pour de plus amples détails.

Les dispositifs d'affichage

Matériel

Les écrans les plus répandus actuellement sont les écrans cathodiques, qui fonctionnent sur le principe de la déflexion d'un jet d'électrons sur un écran recouvert de phosphore. Lorsqu'ils sont frappés par le jet d'électrons, les grains de phosphore émettent de la lumière pendant une certaine durée (appelée temps de rémanence). Le jet d'électrons parcourt l'ensemble de l'écran un certain nombre de fois par seconde, affichant une image ou trame à chaque passage. (Sur un écran de télévision, chaque image est constituée de deux trames et est donc réaffichée en deux passages du jet d'électrons ; cela explique en partie la moins bonne qualité et le moindre prix des écrans de télévision). Une fréquence de réaffichage

plus élevée permet d'utiliser un phosphore avec un temps de rémanence plus faible et d'obtenir une meilleure qualité d'affichage.

Les écrans LCD se répandent de plus en plus, notamment pour les ordinateurs portables, à cause de leur faible consommation électrique et de leur faible encombrement. Ces écrans utilisent les propriétés de polarisation de la lumière par les cristaux liquides. D'autres types d'écrans commencent à apparaître, comme les écrans à plasma dans lequel chaque point est un constitué d'une micro-bulle contenant un plasma dont on peut changer l'état d'excitation.

Écrans à balayage linéaire (écrans bitmap)

La mémoire de rafraîchissement (mémoire écran) contient les valeurs de chaque pixel. Si chaque pixel a un bit, c'est un bitmap, sinon c'est un pixmap (mais on emploie le terme bitmap de manière générique).

Pour une résolution de 1024x1024, on a les valeurs suivantes :

	Mémoire nécessaire	Nombres de couleurs
1 pixel <--> 1 bit	128 Ko	2
1 pixel <--> 8 bits	1 Mo	256
1 pixel <--> 24 bits	3 Mo	16777216 (16 millions)

Dans les systèmes haut de gamme, on peut aller jusqu'à 96 bits par pixels :

- 24 bits de couleur + 8 de fonctions spéciales et de contrôle (alpha channel, etc.)
- double buffer, soit 24+8 bits supplémentaires
- 32 bits pour le Z-buffer (algorithme d'élimination de parties cachées 3D)

Différentes architectures matérielles permettent d'optimiser les performances. En particulier, des processeurs graphiques spécialisés permettent de décharger le CPU en dessinant directement dans la mémoire d'écran.

Table des couleurs

Un grand nombre de bits par pixel permet un plus grand nombre de couleurs, mais coûte plus cher. De plus, la correspondance entre valeur de pixel et couleur est fixe.

La table de couleurs permet de rendre l'usage de la couleur moins cher, et de rendre plus souple la correspondance pixel - couleur. Elle permet également de réaliser des effets spéciaux, par exemple pour l'animation.

La table des couleurs fait correspondre une couleur à chaque valeur possible de pixel. Pour un écran de 8 bits par pixel, la table a 256 entrées. Chaque entrée contient une couleur parmi 16 millions (typiquement). En changeant une entrée de la table des couleurs, tous les pixels correspondants changent de couleur au réaffichage suivant.

Il est souvent utile de modifier les couleurs d'une image sans affecter la structure de l'image elle même, par exemple pour visualiser uniquement certaines parties de l'image ayant une couleur particulière ou créer des effets spéciaux (animation, solarisation,...). Dans ces cas, on utilise des modifications programmées de la table des couleurs.

Les tablettes

Les tablettes graphiques fonctionnent selon différents principes. Elles ont en commun l'utilisation d'un stylet ou d'un "puck", souvent muni de boutons. Leur bonne résolution les rend particulièrement adaptées à des tâches précises ou d'arts graphiques. Certaines tablettes captent, en plus de la position, la pression, voire l'inclinaison du stylet, et même les positions de deux périphériques, par exemple un stylet et un puck tenus dans chaque main. La fréquence d'envoi des informations est de l'ordre de 30 à 60 Hz.

Les tablettes électriques contiennent une grille de fils conducteurs. Des pulsations électriques sont émises séquentiellement dans ces fils et génèrent un signal électrique dans une bobine contenue dans le stylo. La puissance du signal induit est utilisée pour déterminer la position du stylo (et parfois son degré de proximité par rapport à la tablette).

Les tablettes sonores sont munies de 2 ou 3 micros positionnés à la périphérie de la tablette. Le stylo émet des impulsions sonores. Le délai temporel entre l'émission (par le stylo) et la réception (par les micros) est proportionnel à la distance du stylo par rapport à chaque micro (ce mécanisme peut être étendu à trois dimensions : position d'une "chauve-souris" dans l'espace).

Les tablettes résistives utilisent un micro-émetteur de signaux radio haute-fréquence installé dans le stylet. Ce micro-émetteur nécessite d'alimenter le stylet par une batterie ou par un fil électrique. La tablette contient une fine couche d'un matériau conducteur dans lequel un potentiel électrique est créé par les ondes radio. La valeur de ce potentiel électrique aux bordures de la tablette est inversement proportionnelle à la distance.

Les souris, trackball et joysticks

Les souris sont les périphériques d'entrée les plus répandus, grâce à leur faible coût, leur polyvalence et leur faible encombrement. Elles sont le plus souvent mécaniques, parfois optiques. Les souris mécaniques captent la rotation d'une sphère sur laquelle roule la souris, en X et en Y. Les souris optiques utilisent une diode électroluminescente (LED) et une cellule photo-électrique. Elles nécessitent une tablette sur laquelle est gravée une grille. La LED et la cellule permettent de compter le nombre de divisions de la grille qui sont traversées, en X et en Y.

Les trackballs peuvent être décrites comme des souris mécaniques inversées. Elles ont l'avantage d'un encombrement très faible et d'une position fixe. Cependant, la précision est en général moins bonne que celle d'une souris.

Les joysticks ou manches à balai sont munis de ressorts qui ramènent le manche à sa position de repos. Certains joysticks sont à jauge de contrainte : c'est l'effort sur le manche à balai qui est mesuré et non pas son déplacement. Les joysticks contrôlent la vitesse du curseur plutôt que sa position, ce qui introduit une indirection qui peut poser des problèmes à certains utilisateurs. Certains joysticks offrent un 3ème degré de liberté avec la rotation du manche sur lui-même.

Les écrans tactiles et crayons optiques

A la différence des périphériques précédents, les écrans tactiles et stylos optiques permettent une désignation directe sur l'écran des objets affichés. Les écrans tactiles à très faible résolution (10x10 à 50x50) sont à mécanisme optique : une grille de faisceaux infrarouge est interrompue par le doigt lorsqu'il touche l'écran. Les écrans capacitifs ont une résolution de l'ordre de 100x100. Ils détectent le changement d'impédance au toucher. Les écrans sonores utilisent la réflexion d'un signal sonore par le doigt et permettent une résolution de l'ordre de 500x500. Certains écrans tactiles peuvent capter la pression du doigt ou du stylet.

Les crayons optiques, pratiquement inutilisés aujourd'hui, détectent le passage du spot électronique de l'écran par une cellule intégrée au stylo. En mesurant l'intervalle de temps entre le moment où le spot électronique commence le balayage et celui où il passe devant la cellule du crayon, on peut déterminer la position de ce dernier.

Tous ces écrans ne peuvent capter qu'une position à la fois, et donnent des valeurs erronées si l'on pose deux doigts sur l'écran, ou un stylet tenu à la main et le poignet. Il en résulte une fatigue d'utilisation rapide et une faible précision. Enfin, comme l'image est affichée derrière le verre de l'écran et que l'écran tactile est sur le dessus, on a souvent une erreur de parallaxe importante.

Classification des périphériques

Absolu / Relatif

On peut classer les périphériques de localisation en périphériques absolus et relatifs. Les périphériques absolus transmettent une position (x, y) tandis que les périphériques relatifs transmettent un déplacement (dx, dy). Les périphériques relatifs ne permettent pas le dessin à main levée. C'est pourquoi les tablettes graphiques qui sont souvent utilisées en arts graphiques fonctionnent le plus souvent en mode absolu. Par contre, les périphériques absolus ne permettent pas à l'application de repositionner le curseur à l'écran car celui-ci est directement lié à la position du périphérique. Les tablettes, écrans tactiles et stylos optiques sont des périphériques absolus. Les souris, joysticks et trackballs sont relatifs.

Direct / Indirect

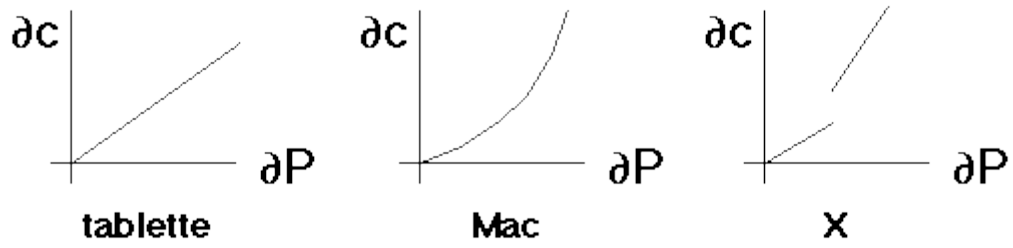
Une autre classification des périphériques de localisation concerne leur degré d'indirection. Un périphérique direct permet un pointage directe sur l'écran (c'est le cas de l'écran tactile et du stylo optique). Un périphérique indirect met en jeu une indirection plus ou moins importante entre l'action sur le périphérique et l'effet sur le curseur. Dans le cas de la tablette ou de la souris, le périphérique se déplace dans un plan horizontal, et contrôle les déplacements du curseur dans le plan de l'écran, généralement vertical. Dans le cas du joystick mécanique, la *position* du périphérique contrôle la *vitesse* de déplacement du curseur. Dans le cas du joystick à jauge de contrainte, c'est l'effort sur le joystick qui contrôle la vitesse de déplacement.

Plus l'indirection est importante, plus l'utilisateur est susceptible d'avoir des difficultés à contrôler le périphérique. Cependant, le pointage direct, s'il est plus naturel, n'a pas que des avantages : le pointage sur écran est fatiguant car il faut tenir le bras en l'air, et la main et le bras occultent une partie de l'écran.

Courbe de réponse

Les périphériques indirects utilisent une courbe de réponse qui transforme les déplacements captés par le périphérique en déplacements du curseur. Dans le cas le plus simple, cette courbe de réponse est linéaire. Ce cas correspond aussi à la courbe de réponse implicite des périphériques directs, comme la tablette.

Dans les schémas ci-dessous, on montre les courbes de réponses de plusieurs périphériques : tablette, souris Macintosh et souris X Window. La courbe montre le déplacement du curseur dC en fonction du déplacement du périphérique dP.



Si l'on veut une bonne précision, il faut que la courbe soit assez plate près de l'origine : à de petits déplacements de la souris correspondent des déplacements encore plus petits du curseur. Par contre, pour une bonne efficacité, il faut pouvoir traverser l'écran rapidement, donc avoir une courbe plutôt raide lorsque dP augmente. Cela conduit à une courbe telle que celle du Macintosh. Lorsque l'on choisit la vitesse de la souris sur le Macintosh, on choisit en fait une courbe de réponse qui monte plus ou moins vite.

La courbe de réponse de X Window est un exemple à ne pas suivre. L'utilisateur peut régler deux paramètres : le seuil, qui est l'abscisse du point de discontinuité de la courbe, et l'accélération, qui est la pente du second segment de droite (le premier segment est de pente fixe). Comme la courbe n'est pas continue, un déplacement de la souris à une vitesse proche du point de discontinuité provoque une réponse erratique du curseur, et cela d'autant plus que le seuil et/ou l'accélération sont importants. De plus, comme les segments sont linéaires, on ne peut pas combiner une bonne efficacité et une bonne précision.

Loi de Fitts (système moteur)

On a vu que temps de pointage d'une cible de taille L à une distance D pouvait être estimé par la loi empirique suivante :

$$T\ (s) = 0.1 \log (2D / L)$$

Voici quelques exemples de valeur pour plusieurs valeurs de L (en colonne) et D (en ligne):

L\D	0,5	2
5	0,30	0,16
20	0,44	0,30

Gestion des entrées

En règle générale, une application graphique interactive dispose de plusieurs périphériques d'entrée (par exemple clavier + souris). Chaque périphérique peut être caractérisé par un état : position et état des boutons de la souris, état de chaque touche du clavier. Les caractéristiques physiques de chaque périphérique déterminent ses états possibles (par exemple, certains claviers ne peuvent avoir qu'une touche appuyée à la fois, à part les touches contrôle, majuscule etc.).

La gestion des entrées consiste à fournir les moyens à l'application de récupérer l'état et/ou les changements d'état des périphériques.

Périphériques logiques

Afin de faire abstraction des particularités des périphériques physiques, certaines librairies introduisent la notion de *périphériques logiques*. Grâce à cette notion, une application peut être rendue pratiquement indépendante de son environnement matériel, et donc plus facilement portable.

GKS et PHIGS distinguent les périphériques logiques suivants :

- *périphérique valeur associée*
- Locator position 2D
- Pick identificateur d'objet affiché à l'écran
- Choice entier représentant le choix dans une liste
- Valuator nombre réel
- String chaîne de caractères
- Stroke séquence de points (tracé)

Dans un environnement donné, ces périphériques logiques seront implémentés par des périphériques physiques ou simulés. Par exemple :

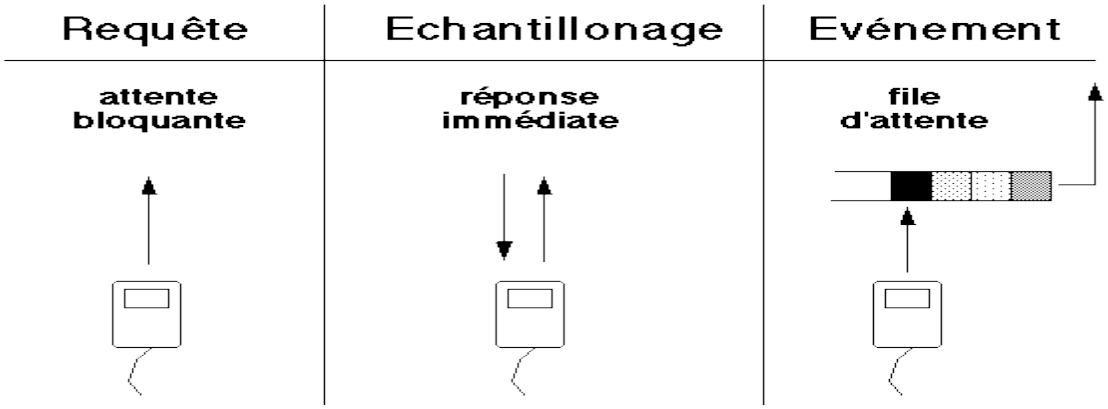
- Locator souris

- Pick souris
- Choice clavier spécial, ou touches fonction du clavier, ou menu à l'écran
- Valuator bouton physique rotatif ou potentiomètre affiché à l'écran
- String clavier (entrée de caractères jusqu'à la touche RETURN)
- Stroke souris (clic - drag - clic)

Les périphériques logiques de GKS et PHIGS correspondent à ce que l'on appellera plus tard des tâches élémentaires d'interaction. Les systèmes graphiques modernes (X Window, Macintosh Toolbox) tendent à ne plus faire de distinctions aussi fines entre les périphériques logiques et restent beaucoup plus proches du niveau physique. En effet, certains périphériques, notamment les périphériques 3D, n'entrent pas aisément dans les catégories ci-dessus. Par ailleurs, ces périphériques logiques préjugent d'un certain style d'interaction que les librairies graphiques modernes ne souhaitent pas imposer. Dans la suite, on ne parle plus de périphériques logiques.

Modes d'entrée

Une librairie graphique peut gérer les entrées de plusieurs manières. Dans les systèmes tels que GKS, il y a 3 modes : requête, échantillonnage, et événement. Dans les systèmes plus récents (X Window, Macintosh Toolbox), le mode principal est le mode événement, avec des possibilités d'échantillonnage.



Dans le mode *requête*, l'application attend un changement d'état d'un périphérique donné. Tant qu'il n'y a pas de changement d'état du périphérique, l'application est bloquée. Ce mode ne permet pas de gérer plusieurs périphériques simultanément.

Dans le mode *échantillonnage*, l'application peut récupérer l'état courant d'un périphérique à tout moment. Cet appel n'est pas bloquant. L'inconvénient de ce mode est qu'il peut conduire à une *attente active* : l'application boucle sur l'échantillonnage d'un (ou plusieurs) périphérique(s) jusqu'à détecter un changement d'état. Cela consomme des ressources CPU inutilement.

Dans le mode *événement*, l'application a accès à une file d'événements qui décrivent les changements d'état des périphériques. Chaque événement contient :

- la date de son occurrence ;
- le périphérique concerné (identificateur de périphérique physique) ;
- l'état (et/ou le changement d'état) du périphérique.

Les événements sont mis dans la file par les couches logicielles basses (pilote de périphérique par exemple). L'application peut éventuellement contrôler quels sont les événements qui l'intéresse afin de ne pas remplir systématiquement la file d'événements inutiles.

L'inconvénient du mode événement est que l'application peut prendre du retard sur les événements, si les événements arrivent plus vite dans la file qu'ils ne sont traités par l'application. La file s'engorge, et les événements ne sont plus forcément traités dans le bon contexte. Idéalement, la file devrait être presque toujours vide.

Exemple :

On se donne deux périphériques, une souris et un clavier. On veut saisir les points d'un polygone de la manière suivante : saisie des points par la souris, et indication de la fin de saisie par le clavier, ou en saisissant deux fois le même point de suite, ou en fermant le polygone (saisie du premier point). On veut également pouvoir annuler l'ensemble de la saisie ou le dernier point entré par le clavier.

Code avec le mode Echantillonage :

```
SampleMouse retourne une structure avec les champs
  button
  position
SampleKbd retourne une structure avec les champs
  key

fonction GetPoly : Poly {
  poly : Poly
  pt1, pt : Point

  poly <- CreerPoly ()

  /* attendre l'appui sur un bouton - ATTENTE ACTIVE */
  tantque non SampleMouse().button
          si SampleKbd().key = FIN
                  retourne poly

  /* enregistrement premier point */
  pt1 <- SampleMouse().position
  AjoutePoint (poly, pt1)

  /* saisir les autres points */
  boucle {
          /* attendre relachement - ATTENTE ACTIVE */
          tantque SampleMouse().button
                  /* rien */
          /* attendre prochain point ou touche fin */
          tantque non SampleMouse().button
                  cas SampleKbd().key {
                          FIN: retourne poly
                          ANNULE: EnlevePoint (poly)
```

```
                        ANNULETOUT: EnleveTout (poly)
                }
        /* saisir le point */
        pt <- SampleMouse().position
        si pt = pt1
                retourne poly
        sinon
                AjoutePoint (poly, pt)
  }
}
```

Code avec le mode Evenement:

```
WaitEvent retourne une structure Event :
Event = struct
  type : ButtonUp, ButtonDown, Key
  cas device : (Mouse, Kbd) {
        Mouse : button : bool; pos : Point
        Kbd : key : caractere
  }

fonction GetPoly : Poly {
  ev : Event
  fin : bool <- faux
  premier : bool <- vrai
  poly : Poly
  pt1, pt : Point

  poly <- CreerPoly ()
  tantque non fini {
        ev <- WaitEvent ()
        cas ev.type {
                ButtonDown:
                        /* saisir un point */
                        pt <- ev.position
                        si premier {
                                pt1 <- pt
                                premier <- faux
                                AjoutePoint (poly, pt)
                        } sinonsi pt1 = pt
                                fini <- vrai
                        sinon
                                AjoutePoint (poly, pt)
                Key:
                        /* traiter une touche clavier */
                        cas ev.key {
                                FIN : fini <- vrai
                                ANNULE : EnlevePoint (poly)
                                ANNULETOUT : EnleveTout(poly)
                        }
        }
  }
  retourne poly
}
```

Remarque : on ne peut pas écrire ce programme avec le mode Requête, car ce mode interdit d'attendre sur deux périphériques simultanément.

Exercice :

On suppose que, en mode événement, les seuls changements d'état de la souris qui provoquent un événement sont les changements d'état des boutons. Comment peut-on alors implémenter la saisie de la trajectoire de la souris entre un appui et un relâchement de bouton (pour implémenter par exemple de la reconnaissance de geste) ?

Note : on peut donner un "timeout" à la fonction WaitEvent : si la file est vide et qu'aucun événement n'arrive dans l'intervalle de temps spécifié, la fonction retourne un événement de type TimeOut, indiquant qu'il ne s'est rien passé.

Exercice :

On suppose maintenant que les changements de position de la souris génèrent des événements. Comment faire pour suivre au plus près les déplacements de la souris, c'est-à-dire éviter l'engorgement de la file d'événements ?

Note : On se donne une fonction qui retourne le nombre d'événements dans la file.

Echo

L'écho est la représentation graphique des actions de l'utilisateur sur les périphériques logiques. Pour la souris, l'écho le plus connu est le *curseur*, un icone (souvent une flèche) qui "suit" les mouvements de la souris.

Les librairies graphiques permettent parfois un contrôle de l'écho des périphériques.

Pour la souris, on peut disposer des modes d'écho suivants :

- curseur - on peut spécifier la forme du curseur ;
- ligne élastique (rubber_line) - on spécifie le point d'ancrage ; une ligne est dessinée entre le point d'ancrage et la position courante de la souris ;
- rectangle élastique (rubber_rect) - on spécifie le point d'ancrage ; un rectangle est dessiné entre le point d'ancrage et la position courante de la souris.

Pour les modes "élastiques", l'écho est en général tracé en mode XOR pour pouvoir restaurer le contenu de l'écran facilement.

Exercice :

Ecrire une procédure de saisie d'un segment (appuyer - déplacer - relâcher) avec écho du segment en cours de définition

- en utilisant un écho de type ligne élastique
- en générant l'écho "à la main".

Machines à états

Dans une interface graphique, l'utilisateur doit pouvoir accomplir un certain nombre de *tâches* telles que l'édition d'un rapport, le calcul d'un bilan, etc. Ces tâches peuvent être décomposées en sous-tâches plus simples (déplacer un paragraphe, saisir une formule, etc.). Cette décomposition se termine avec des tâches élémentaires et composées que nous allons décrire.

Une technique utile pour décrire ces tâches consiste à utiliser des *machines à états.* Une machine à états est constituée d'un automate à états finis, à savoir :

- un ensemble d'états E
- un état initial e_0
- un sous-ensemble F d'états finaux (F inclus dans E)
- un ensemble de transitions T, définissant une fonction de E x V -> E, V étant un alphabet fini.

A tout instant, l'automate est dans un état e (au départ, e = e_0). Il lit un caractère c de l'alphabet et avance vers l'état T (e, c).

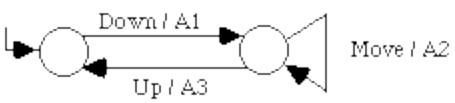
En interaction graphique, l'alphabet est constitué des types d'événements possibles (par exemple ButtonDown, ButtonUp, MouseMove, KeyDown, KeyUp).

Une machine à états est un automate auquel on ajoute des *actions*. Selon le type de machine, on peut associer l'action à chaque transition ou à chaque état.

- action associée à un état : elle est exécutée lorsque cet état devient l'état courant ;
- action associée à une transition : elle est exécutée lorsque cette transition est activée.

L'association des actions aux transitions est plus puissante et donne plus de souplesse.

Exemple : segment élastique



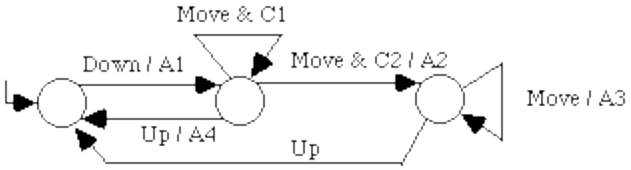
Actions :

```
A1 : P1<-event.position ; P2<-P1
A2 : Effacer (P1,P2) ; P2<-event.position ;  Dessiner (P1,P2)
A3 : Effacer (P1, P2) ; Créer (P1, P2)
```

Ce type de machine à états est rapidement limité car les transitions ne dépendent pas seulement de l'événement, mais aussi d'un état global qu'il n'est pas toujours possible (ou commode) de représenter par des états de l'automate. On peut ajouter des *conditions* aux transitions. On parle alors de *réseau prédicat-transition*. Pour que le réseau soit déterministe, il faut que les conditions qui portent sur des transitions issues du même état avec le même événement soient mutuellement exclusives.

Exemple : déplacement/sélection d'objet avec hystérésis

Un clic sur un objet le sélectionne, tandis qu'un "drag" le déplace. Pour prendre en compte le fait que le clic génère souvent un déplacement minime de la souris, on introduit un hystérésis : le déplacement ne doit commencer que lorsque l'on a bougé la souris suffisamment. Si l'on relâche sans avoir bougé suffisamment pour déclencher un déplacement, on sélectionne l'objet.



Actions et conditions :

```
A1 : P0 <- event.position ; obj <- Objet (P0)
C1 : Distance (P0, event.position ) < minDist
C2 : not C1
A2 : P1 <- event.position ; Déplace (obj, P1 - P0); P0 <- P1
A3 : A2
A4 : Selectionne (obj)
```

Cet automate ne prend pas en compte ce qui se passe si l'on clique en dehors d'un objet (dans A1, on suppose que obj != nul). Pour cela il faudrait ajouter une condition dans la transition Down/A1 : s'il n'y a pas d'objet sous la souris, passer dans un mode de sélection par rectangle, représenté par une autre partie de l'automate.

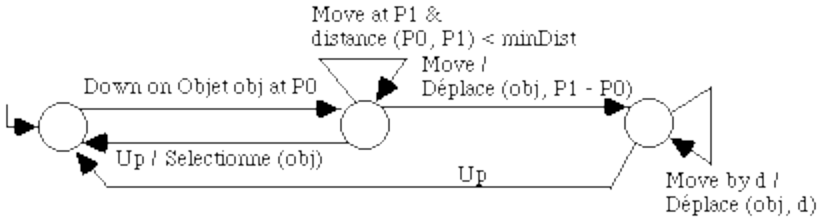
Pour simplifier l'écriture des conditions, on peut utiliser des transitions de la forme (<event> est un type d'événement, <type> un type d'objet et <var>, <pos>, <delta> des noms de variables :

```
<event> [on <type> [<var>]] [at <pos>] [by <delta>]
```

- <event> on <type> <var> est équivalente à une transition où la condition teste si la position de l'événement correspond à un objet de type <type> et où l'objet désigné est accessible par la variable <var> dans l'action associée.
- <event> at <pos> est équivalent à une transition où l'action associée stocke la position de l'événements dans la variable <pos>.
- <event> by <delta> est équivalent à une transition où l'action associée stocke le déplacement depuis le dernier événement dans la variable <delta>.

De plus, si plusieurs transitions sont issues d'un même état par le même événement, on autorise que l'une d'entre elles n'ait pas de condition. On considère alors que cette transition n'est franchie que lorsqu'aucune des conditions des autres transitions n'est satisfaite.

L'automate ci-dessus peut alors s'écrire :



L'implémentation d'une machine à états est simple. Il suffit d'une variable globale qui représente l'état courant. Le schéma général est le suivant :

```
event <- NextEvent ();
cas event.type {
  ...
  type_i :
          cas état {
                  ...
                  état_j :        si Cij1 {
                                          Aij1; état <- état_ij1
                                  } sinon
                                  ...
                                  si Cijk {
                                          Aijk ; état <- état_ijk
                                  } sinon
                                  ...
                  ...
          }
  ...
}
```

Dans la pratique, on découpe ce code en morceaux, par exemple en écrivant une procédure pour chaque type d'événement (HandleUp, HandleDown, etc.).

Problèmes de ces machines à états :

- l'automate peut devenir gros ;
- il est difficile de traiter correctement les erreurs ;
- il n'y a pas de modularité. Il est donc difficile de réutiliser des machines existantes ;
- les sorties ne sont pas formalisées : elles sont gérées par effet de bord via les actions.

D'autres modèles de machines à états, plus puissants, essaient de pallier ces inconvénients. Nous n'en parlerons pas car le modèle ci-dessus suffit pour les tâches que l'on va étudier.

Tâches de la manipulation directe

Le terme de manipulation directe a été introduit pour la première fois en 1983 par Ben Shneiderman. Dans les interfaces à manipulation directe, l'utilisateur à l'impression d'agir directement sur un monde qui s'apparente à un monde physique. Plutôt que de communiquer avec un interlocuteur auquel il demande des informations, ou ordonne d'exécuter des actions, l'utilisateur a l'impression que c'est lui-même qui agit sur des objets dotés de réactions spécifiques.

Principes de la manipulation directe

Les principes de la manipulation directe qui ont été énoncés en 1983 par Shneiderman sont :

- Représentation continue des objets ;
- Utilisation d'actions physiques (mouvement et sélection par la souris, pointage , etc.), au lieu de syntaxe complexe ;
- Opérations rapides, incrémentales et réversibles, dont les effets sur les objets doivent être visibles immédiatement ;
- Apprentissage selon une approche progressive afin de permettre l'utilisation de l'interface même avec un minimum de connaissances.

Ces principes impliquent que l'état du système doit être présenté à l'utilisateur de façon permanente par l'intermédiaire d'objets pouvant être manipulés directement par des actions physiques. Les effets de ces actions doivent être visibles immédiatement sur les objets concernés. Les actions doivent être réversibles c'est-à-dire que leurs effets peuvent être annulés afin de remettre le système dans son état antérieur. Cette caractéristique encourage l'utilisateur à expérimenter le système en osant exécuter des actions dont les effets lui sont inconnus. Ceci permet donc d'assurer un apprentissage progressif et au rythme de l'utilisateur.

Utilisation de la manipulation directe

L'usage d'un périphérique de localisation est primordial pour la manipulation directe, car c'est lui qui permet d'agir directement sur les objets.

Les interactions élémentaires dans un système muni d'une souris (ou d'une tablette) sont les suivantes :

- **pointage**

Déplacer le curseur sur une zone de l'écran.

- **sélection**

Appuyer puis relâcher le bouton de la souris (clic). L'utilisation simultanée de touches du clavier (shift, control, option, alt, etc.) et/ou de clicks multiples permet de distinguer différentes actions de sélection.

- **tracé**

Appuyer sur le bouton de la souris, déplacer le curseur puis relâcher le bouton (cliquer-tirer ou "drag and drop"). Comme pour la sélection, l'usage simultané de touches de clavier permet de distinguer plusieurs actions de tracé.

Les caractéristiques du dispositif de pointage affectent la précision et la rapidité de ces interactions élémentaires. Ainsi, le temps de pointage est donné par la Loi de Fitts. Pour une cible de diamètre L situé à une distance D du curseur, le temps de pointage de la cible est de l'ordre de (en secondes) :

t = 0.1 Log 2D/L

Pour une bonne précision, la résolution du périphérique doit être égale ou supérieure à celle de l'écran. Une courbe de réponse non linéaire entre les mouvements du périphérique et ceux du curseur permet de réaliser un bon compromis entre vitesse et précision dans le cas des périphériques relatifs (souris, trackball) : voir le réglage de l'accélération de la souris sur le Macintosh et sous X Window, par exemple.

L'interaction de tracé peut interférer avec la sélection : lorsque l'on fait un clic, la souris peut bouger pendant que le bouton est enfoncé, et déclencher une interaction de tracé. Pour éviter cela, on prévoit généralement un *hystérésis* : l'interaction de tracé n'est reconnue après l'appui sur le bouton que lorsque le curseur a bougé suffisamment.

Les tâches interactives que l'on peut réaliser par manipulation directe sont :

- Saisie de valeurs (texte, quantité, position, tracé, etc.) ;
- Sélection d'un ou plusieurs objets parmi un ensemble ;
- Déclenchement de commandes ;
- Défilement de documents ;
- Spécification d'arguments et de propriétés ;
- Transformations graphiques.

Dans la suite, nous décrivons comment ces tâches interactives élémentaires sont réalisées dans les interfaces graphiques actuelles à partir des trois techniques d'interaction de base.

Tâches de saisie

Les tâches de saisie sont destinées à entrer des informations que le système ne peut présenter simplement dans un ensemble prédéfini : nom d'un nouveau fichier, valeur numérique, position à l'écran, etc.

Saisie de textes

Une boîte de saisie permet à l'utilisateur d'entrer et d'éditer du texte. Il en existe plusieurs sortes : ligne unique, multiligne, conversion automatique des minuscules en majuscules, masquage de l'affichage (pour les mots de passe). Les boîtes de saisie multilignes peuvent disposer de barres de défilement.

Saisie de quantités

Différentes techniques sont possibles :

- taper la valeur au clavier,
- positionner l'aiguille d'un compteur ou d'un potentiomètre rotatif,
- positionner le curseur d'un potentiomètre ,
- incrémenter, décrémenter un compteur à l'aide de deux boutons (haut, bas).

La technique à employer dépend de l'ensemble et du type des valeurs possibles : valeurs continues ou discontinues, nombre de valeurs, taille de l'intervalle, etc.

Saisie de positions

Des positions à l'écran peuvent être saisies par pointage et validation par la souris.

La saisie d'un rectangle se fait par une interaction de type cliquer-tirer : on appuie sur le bouton de la souris, un rectangle de sélection apparaît et suit la souris jusqu'au relâchement. Les deux points saisis correspondent à une diagonale du rectangle. De la même façon, la saisie de formes élastiques peut être utilisées pour saisir des cercles, ellipses, etc. par une interaction de type cliquer-tirer.

Lorsque l'on doit saisir plusieurs positions successives (par exemple les sommets d'un polygones), un feed-back doit permettre de voir les positions précédemment saisies et il doit être possible d'annuler la dernière saisie ou la saisie complète. L'indication de la fin de saisie d'une suite de points est un problème délicat. Aucune des solutions ci-dessous n'est idéale :

- double-clic ;
- saisie (proche) du premier point (cas d'un contour fermé) ;
- touche de validation.

Saisie de tracés

La saisie de tracé consiste à échantillonner les positions de la souris lors d'une interaction de type cliquer-tirer. Pour des applications dans le domaine graphique (retouche photographique par exemple), on préfère utiliser une tablette qui est plus précise et correspond mieux aux habitudes des graphistes. Certaines tablettes fournissent une indication de pression qui peut être échantillonnée en même temps que la position et enrichir le tracé. Par exemple, la pression peut être interprétée comme l'épaisseur du trait.

Un feed-back lexical (l'encre) permet de visualiser le tracé. Cette encre disparaît à la fin de l'interaction ("lever de stylo" si l'on utilise une tablette au lieu d'une souris) et est remplacée par le résultat de la saisie.

La saisie de tracé peut par ailleurs servir à la sélection par entourage "au lasso" des éléments à sélectionner.

Tâches de sélection

La sélection est l'opération qui consiste à choisir un élément (ou plusieurs) parmi un ensemble d'éléments. On peut distinguer plusieurs types de sélections selon que l'ensemble des éléments est de cardinal fixe (ou en général fixe) ou fréquemment variable, selon le nombre d'éléments et selon que l'on veut pouvoir sélectionner un seul élément à la fois (sélection simple) ou plusieurs (sélection multiple).

Les éléments de l'ensemble de choix sont en général des commandes, des attributs, des types d'objets ou des objets. Le cardinal de l'ensemble est fréquemment variable dans ce dernier cas (création, destruction des instances de manière dynamique de la part de l'utilisateur). Il est en général constant dans les autres.

La sélection dans un ensemble variable de choix peut se faire par :

- sélection directe par pointage ;
- sélection dans une liste ;
- saisie d'un nom.

La sélection dans un ensemble fixe de choix peut se faire par :

- sélection dans un menu ;
- boutons radio ;
- cases à cocher.

Sélection directe par pointage

La sélection directe par pointage (ou *désignation*) consiste à spécifier un objet présent à l'écran en le pointant avec la souris et en validant par un clic. Cette spécification se fait à l'aide de la souris et éventuellement en utilisant les touches de contrôle du curseur (flèches).

Pour aider le positionnement, plusieurs techniques peuvent être utilisées :

- **Feed-back**
  la cible (l'objet couramment sous le curseur) peut être mise en évidence par une mise en sur-brillance, ou par le changement de forme du curseur. Par exemple, dans un éditeur de dessin, le curseur a une forme différente selon que l'on est dans un objet, sur son bord, ou à l'extérieur. Autre exemple : un bouton "s'éclaire" lorsque le curseur passe dessus pour montrer qu'il est sensible et peut être désigné.
- **Gravité**
  Le pointage d'objets de petit taille (segments de droite par exemple) peut être facilité en ajoutant une tolérance lors du pointage et de la sélection : l'objet est désigné lorsque le curseur est dans son *champ de gravité*.

Après sélection, l'objet ou les objets sélectionnés sont souvent mis en évidence (changement de couleur, inversion vidéo, ajout de "poignées", etc.) car la sélection est souvent utilisée comme paramètre implicite des commandes (par exemple, couper/copier).

Parfois il existe différents niveaux de granularité pour la sélection (par exemple dans un traitement de textes, caractère, mot, ligne, paragraphe...). On peut alors soit utiliser la même opération de pointage et dans ce cas prévoir des commandes de changement de niveau de granularité, ou bien utiliser des opérations de pointage différentes (clic pour le caractère, double clic pour le mot, utilisation des modifieurs - shift, alt, ctrl - combinés au clic, etc.). La première solution n'est pas efficace si on change de granularité fréquemment.

La seconde pose problème si les niveaux de granularité sont nombreux ou évoluent de manière dynamique.

Sélection multiple

La sélection de plusieurs éléments peut se faire :

- par groupe : sélection par rectangle, lasso ...
- par ajout/retrait dans un ensemble de sélection : "shift-clic"

Sélection dans une liste

Une liste permet à l'utilisateur de choisir un élément (ou plusieurs) dans une liste d'éléments. Les listes sont en général accompagnées d'une barre de défilement. La plupart des listes contiennent du texte, cependant il est possible d'y mettre de petites images. Parfois, le contenu d'une liste est hiérarchisé : on peut ouvrir ou fermer des sous-sections de la liste.

Sélection multiple

La sélection de plusieurs éléments peut se faire :

- par intervalle : sélection du premier puis du dernier élément ;
- par ajout/retrait dans un ensemble de sélection : "shift-clic", ou bien une seconde liste et 2 boutons permettant d'ajouter et retirer des éléments.

Sélection par saisie d'un nom

Cette technique est pratique si l'utilisateur est sensé connaître tous les identificateurs des éléments de l'ensemble (ou ceux qui l'intéressent en général) et si ces identificateurs ne sont pas trop longs (exemple : choix d'une machine sur laquelle on désire se connecter). De façon générale, il est beaucoup plus facile de retrouver un nom dans une liste que de s'en souvenir pour pouvoir le taper.

Combinaison de techniques

Une technique de sélection fréquemment employée consiste à proposer à l'utilisateur une boîte de saisie dans laquelle il peut taper le nom de l'élément ainsi qu'une boîte à liste contenant tous les éléments triés par ordre alphabétique. Au fur et à mesure que l'utilisateur tape le nom de l'élément, le curseur de la boîte à liste se positionne automatiquement sur le premier élément commençant par les lettres tapées. Ainsi, l'utilisateur peut combiner la sélection par saisie et la sélection par pointage pour augmenter la rapidité d'accès. (Exemple : recherche d'une page d'aide par mot-clé du glossaire).

Sélection par menu

Les menus servent soit à déclencher des commandes (Couper, Enregistrer, etc.), soit à sélectionner un état (palette d'outils, police de caractère, etc.). On peut distinguer trois sortes de menus : les menus fixes (palettes) et les menus surgissants (pop-up) ou déroulants (roll-down) .

Les palettes occupent une position fixe à l'écran. Les pop-ups peuvent surgir n'importe où, en général à la position courante du curseur de la souris et suite à un événement particulier (appui du bouton droit de la souris sur un objet particulier). Ces derniers présentent l'avantage de ne pas occuper d'espace à l'écran et d'être en général contextuels. En effet, le menu affiché peut dépendre du type de l'objet sur lequel s'est produit le clic. Dans ce cas, le menu ainsi affiché présente les propriétés de l'objet en question (et permet de les modifier) et/ou les commandes exécutables sur ce type d'objet. Les menus déroulants apparaissent à des positions prédéfinies comme par exemple la barre de menus.

Le nombre de rubriques par menu doit être limité : 7 +/- 2 s'il s'agit d'une liste non homogène. Pour une liste homogène (polices de caractères, fenêtres), la liste peut être plus longue. Dans ce cas, elle doit être ordonnée et l'on peut mettre une section en tête du menu avec les sélections les plus récentes ou les plus fréquentes. Si un menu ne peut pas être totalement affiché sur l'écran par manque de place, il faut prévoir un mécanisme de défilement.

Les rubriques de menus ne sont pas forcément du texte; elles peuvent être des images (bitmaps de petit taille, style de traits par exemple). D'autre part certains menus peuvent changer de manière dynamique (par exemple, liste des derniers fichiers utilisés ou des fenêtres ouvertes).

L'interaction avec les menus peut être améliorée par l'usage de plusieurs techniques :

- **Feed-back**
  Il est important d'indiquer clairement à l'utilisateur la rubrique courante du menu (inversion vidéo), ainsi qu'éventuellement la rubrique sélectionnée (par exemple dans une palette d'outils).
  Il est également important de montrer à l'utilisateur le chemin parcouru dans l'arborescence des menus (en maintenant les rubriques en question en inverse vidéo).
- **Séparateurs**
  Les rubriques de menus, peuvent être divisées en groupe en utilisant des séparateurs (traits horizontaux). Ceci permet de les classer selon leur rôle fonctionnel et permettre ainsi une meilleure mémorisation et une sélection plus rapide.
- **Rubriques grisées**
  Une rubrique grisée signifie que celle-ci n'est pas active et qu'elle ne peut donc pas être sélectionnée (par exemple : *coller* si le presse-papier est vide ou*défaire* si la dernière opération ne peut pas être annulée.)
- **Mnémoniques et raccourcis clavier**

Le mnémonique correspond à la lettre qui est en général soulignée dans le texte de la rubrique. Il signifie que cette rubrique peut être sélectionnée en appuyant simultanément sur un modifieur (touche Alt ou autre) et sur la touche correspondant au caractère souligné. Cependant ceci n'est valable que lorsque la rubrique est visible. Les raccourcis clavier apparaissent généralement à côté de la rubrique et permettent de sélectionner une rubrique d'un menu même si elle n'est pas visible. Ils permettent d'avoir un accès direct à la rubrique (sans avoir à parcourir l'arborescence des menus).

Il existe de nombreuses variantes ou extensions des menus décrits ci-dessus. Parmi les plus importantes on peut citer :

- **Menus hiérarchiques**
  Les menus sont parfois définis par une structure de données hiérarchisée. La barre de menus peut être considérée comme un premier niveau de menus. Un menu est composé d'items ou rubriques. Une rubrique peut être elle-même l'entête d'un sous-menu. Dans ce cas, on fait suivre en général le texte de la rubrique par un symbole (flèche >) indiquant à l'utilisateur que cette rubrique donne accès à un sous-menu. L'accès à un sous-menu peut être explicite (clic) ou implicite (dès que le curseur de la souris passe sur l'entête du menu, celui-ci est déroulé). Il convient de ne pas abuser des niveaux de hiérarchie pour une meilleure mémorisation (3 niveaux au maximum).
- **Menus détachables**
  Il est parfois possible de transformer un menu fugitif (surgissant ou déroulant) en palette, par exemple en le détachant de son titre dans la barre de menu (Macintosh) ou en sélectionnant le premier séparateur (Tk). Cela permet un accès plus facile aux rubriques du menu si on en a un besoin fréquent. Pendant qu'il est détaché, le menu fugitif reste accessible. Une fois détaché, le menu peut être refermé.
- **Menus circulaires**
  Dans un menu circulaire, les rubriques sont réparties en secteurs. Le nombre maximal de rubriques est de 8 à 10. L'avantage d'un menu circulaire est que la distance par rapport à chaque item est la même et donc le temps de sélection ne dépend pas de la position dans le menu. Comme les menus linéaires, les menus circulaires peuvent être hiérarchiques.
  "Marking menus"
  Les "marking menus" sont une variante des menus circulaires qui permet de sélectionner une rubrique par un geste rapide avant que le menu n'apparaisse.

Sélection exclusive : boutons radio

Les boutons radio permettent de sélectionner une option parmi un ensemble d'options exclusives : un seul des boutons du groupe peut être activé à un instant donné (comme les bandes de fréquence sur les anciens modèles de radios).

Sélection binaire : cases à cocher

Une case à cocher peut être assimilée à un interrupteur à deux états : allumé ou éteint, (ou à un groupe de 2 boutons radio). Elle sert à sélectionner un état parmi deux possibles, en général de forme on/off, activé/désactivé, etc.

Tâches de déclenchement

Les tâches de déclenchement servent à activer des commandes. Nous avons déjà vu que les rubriques d'un menu peuvent être des commandes, déclenchées par sélection dans le menu. Cette section présente d'autres techniques pour réaliser les tâches de déclenchement.

Boutons

Un bouton est une zone de l'écran que l'on peut activer par pointage et sélection. En général l'action du bouton est déclenchée sur le relâchement du bouton de la souris, ce qui permet d'annuler une action : après avoir appuyé sur le bouton (de la souris) dans le bouton (à l'écran), on déplace le curseur hors du bouton (à l'écran) et on relâche le bouton (de la souris).

De la même façon que les rubriques d'un menu peuvent être grisées lorsqu'elles correspondent à des commandes invalides, on peut griser un bouton lorsque sa commande associée est invalide.

Le contenu d'un bouton peut être du texte et/ou une image.

On peut considérer un menu de commandes ou une palette comme un ensemble de boutons.

Cliquer-Tirer ("Drag and drop")

L'interaction de cliquer-tirer consiste à prendre un objet source désigné par pointage lors de l'enfoncement du bouton de la souris et à le déplacer vers un objet cible désigné par pointage lors du relâchement du bouton. La trajectoire utilisée entre la source et la cible n'est pas prise en compte. A la fin de l'action, une commande qui dépend de la source et de la cible est déclenchée.

Pendant l'interaction, le feed-back lexical consiste à afficher la forme de l'objet source (ou son "ombre") en suivant les déplacements de la souris. Un feed-back syntaxique ou sémantique permet d'indiquer quels sont les objets cibles possibles : lorsque la souris passe sur une cible possible, celle-ci change d'aspect (inversion vidéo, sur-brillance, etc.).

Le cliquer-tirer peut déclencher un défilement automatique (voir plus loin) lorsque la cible n'est pas visible dans une fenêtre. Le cliquer-tirer peut aussi être utilisé entre des fenêtres, c'est-à-dire que la source et la cible peuvent être dans des fenêtres différentes.

Entrée gestuelle

La saisie de tracé peut être utilisée pour déclencher des commandes, dites commandes gestuelles. Dans ce cas il est préférable d'utiliser un stylet pour la saisie, soit sur une tablette graphique, soit sur un écran tactile, car la précision est meilleure. Un feed-back lexical (l'encre) permet de donner l'illusion d'un crayon électronique. Ce feed-back disparaît au lever du stylo (ou au relâchement du bouton).

La commande déclenchée dépend de la forme du tracé. Cette forme doit être interprétée et reconnue par rapport à un vocabulaire prédéfini. De cette forme sont extraits la commande à exécuter et ses paramètres (taille, position, etc.)

Tâches de défilement

Très souvent, les informations à présenter ne peuvent être affichées dans l'espace disponible : texte d'un document, liste, long menu, etc. Il faut donc pouvoir accéder aux parties non visibles. Dans cette section, nous appelons "document" l'ensemble des informations à rendre accessibles par le défilement. Il peut s'agir d'un document au sens habituel du terme (texte, dessin, etc.) mais aussi d'une liste de noms, d'un ensemble d'icones, etc.

Barres de défilement

Le défilement est le plus souvent réalisé par une ou deux barres de défilement : l'ascenseur indique la position relative de la partie visible par rapport à l'ensemble des informations et peut être manipulé directement. Les flèches permettent un déplacement incrémental dont la granularité dépend des informations : ligne par ligne pour le texte, pixel par pixel pour un dessin, etc. Parfois, il est possible de faire défiler les informations par page entière.

Les barres de défilement posent un problème : dans quel sens doit défiler le contenu de la fenêtre lorsque l'on clique sur la flèche vers le haut ? Puisque l'ascenseur représente la position relative dans le document, il est normal que lorsque l'ascenseur est en haut, on soit au début du document (dans l'exemple du défilement d'un document texte dans une fenêtre). En conséquence, lorsque l'on déplace l'ascenseur *vers le haut*, le contenu de la fenêtre défile *vers le bas*, laissant apparaître la section précédente du document. Pour être cohérent avec ce fonctionnement, la flèche *vers le haut* contrôle le déplacement de l'ascenseur, et fait donc défiler le texte *vers le bas*...

Défilement direct

Une façon plus intuitive de contrôler le défilement est d'agir directement sur le document à faire défiler par une interaction de type tracé : le document suit les déplacements de la souris, de telle sorte que la position relative du curseur par rapport au document reste la même (mode "main" des outils de dessin par exemple). De plus cela permet d'économiser l'espace de l'écran puisque les barres de défilement deviennent inutiles.

Inconvénients :

- il peut être fastidieux de faire défiler de longs documents, à moins d'utiliser le défilement automatique (ci-dessous) ;
- l'action directe sur le document (pointage ou tracé) est généralement utilisée pour l'édition du document, et il faut donc que le défilement direct soit un mode accessible par une palette ;
- le défilement direct ne permet pas de connaître la position relative dans le document.

Défilement automatique

Dans certains cas, le défilement peut être automatiquement déclenché en fonction de la position du curseur. Par exemple, si un menu est trop long pour être affiché, on présente un sous-ensemble des rubriques et le déplacement du curseur au début ou à la fin du menu déclenche le défilement. De la même façon, lorsque l'on fait défiler un document par défilement direct ou lors d'un cliquer-tirer, on peut déclencher le défilement automatique lorsque le curseur sort de la fenêtre.

Pour faciliter le contrôle par l'utilisateur, la vitesse de défilement doit être progressive. Une possibilité est d'augmenter la vitesse tant que le curseur est immobile et de la réduire lorsqu'il bouge.

Tâches de spécification d'arguments et de propriétés

L'interaction par manipulation directe consiste à activer des commandes par action directe sur des éléments graphiques présents à l'écran. La spécification des arguments de ces commandes peut se faire de plusieurs façons :

- directement : l'action spécifiant la commande inclut également les arguments (comme par exemple dans l'entrée gestuelle) ;
- implicitement : un ensemble distingué d'objets, appelé "sélection" sert d'argument implicite à la commande ;
- explicitement : une *boîte de dialogue* permet de spécifier les arguments complémentaires.

Le principe des boîtes de dialogue est également utilisé pour la notification par l'application d'erreurs ou pour la présentation d'informations. Bien qu'il ne s'agisse pas à proprement parler de tâches de l'utilisateur, les *boîtes d'alerte et d'information* sont présentées dans cette section.

Enfin, les *boîtes de propriétés* sont utilisées à la fois pour présenter des informations sur les objets (appelées propriétés) et permettre leur modification.

Les boîtes de dialogue sont des fenêtres grâce auxquelles un programme peut demander à l'utilisateur les informations nécessaires à l'exécution d'une commande. Elles apparaissent en général à la suite de la sélection d'une commande, typiquement dans un menu.

Les boites de dialogue contiennent en général au moins 2 boutons :

- OK : validation
- CANCEL : annulation

Parfois on peut trouver un bouton APPLY qui permet de d'appliquer les nouvelles valeurs sans avoir à fermer la boîte de dialogue. Dans ce cas la boîte de dialogue se rapproche d'une boîte de propriétés.

Pour faciliter l'interaction de l'utilisateur, il est important de choisir des valeurs par défaut judicieuses pour les différents champs de la boîte de dialogue. Cela permet de limiter le nombre d'interactions nécessaires avant la validation. Par exemple, lorsque l'on active la commande "Enregistrer sous", le nom par défaut du fichier peut être le nom courant plutôt qu'une chaîne vide ou prédéfinie ("Sans titre").

Boîtes modales et non modales

On peut distinguer deux types de boîtes de dialogue : les boîtes de dialogue modales, et les boîtes de dialogue non modales (la notion de mode sera décrite plus loin). Une boîte de dialogue modale empêche l'utilisateur d'accéder à toute autre fenêtre de l'application tant qu'il ne l'a pas refermée (soit par annulation, soit par validation). Dans ce cas, les informations requises par la boîte de dialogue sont en général importantes. A l'inverse, une boîte de dialogue non modale autorise l'utilisateur à interagir avec d'autres fenêtres sans fermeture de la boîte de dialogue.

Les boîtes de dialogue modales peuvent interdire l'accès soit seulement aux autres fenêtres de l'application (modale locale), soit à toutes les fenêtres de l'écran (modale globale). Ce dernier types de boîtes de dialogue est à éviter en général sauf dans le cas ou les informations contenues dans la boîte concernent tout le système.

Parties optionnelles

Lorsque certains champs de la boîte de dialogue ne sont pas indispensables, ils peuvent être mis dans une partie optionnelle que l'on peut ouvrir ou fermer. Lors de la prochaine ouverture de la boîte de dialogue, il est préférable de retrouver la partie optionnelle dans l'état où on l'avait laissée (ouverte ou fermée).

Rubriques

Lorsque le nombre de champs est très important, on peut les regrouper en rubriques. Chaque rubrique est accessible par un onglet ("tab") ou un menu déroulant. Cette technique est souvent utilisée pour la définition des préférences.

Un des problèmes principaux des boîtes à onglets est que la notion de validation/annulation n'est pas toujours claire : est-ce que les modifications sont prises en compte ou sont annulées globalement lorsque l'on ferme la boîte de dialogue ou bien à chaque fois que l'on change d'onglet ?

Boîtes d'alerte et d'information

Au contraire des boîtes de dialogue, les boîtes d'alerte apparaissent de façon non sollicitée par l'utilisateur, soit en cas d'erreur soit pour l'informer de l'état du système. En général, ce sont des boîtes modales (locales ou globales). Dans le cas le plus simple, un message d'erreur ou d'information est affiché avec un bouton "OK" permettant de faire disparaître la boîte.

Boîtes de progression

Un cas particulier de boîte d'information sont les boîtes de progression, qui indiquent l'état d'avancement d'une action par le système. Ces boîtes apparaissent (ou devraient apparaître) dès qu'une action du système est longue, c'est-à-dire supérieur à une ou deux secondes.

Boîtes de propriétés

Les boîtes de propriétés permettent d'afficher en permanence à l'écran les propriétés d'autres objets, comme par exemple la position courante du curseur, les attributs typographiques d'un texte, les attributs graphiques d'un élément de dessin ou les paramètres d'un outil d'édition. Les boîtes de propriétés sont similaires aux palettes, sauf que les palettes contiennent à priori un ensemble de rubriques de même nature (outils, polices de caractères, couleurs, etc.) alors qu'une boîte de propriétés contient des champs de natures diverses.

Contrairement aux boîtes de dialogue, les boîtes de propriétés n'ont pas de bouton de validation : toute action sur un champ a un effet immédiat sur le ou les objets concernés. Inversement, toute action sur l'objet concerné (ou la sélection d'un objet différent) met à jour les champs de la boîte de propriétés.

Comme les boîtes de dialogue, les boîtes de propriétés peuvent avoir des rubriques et/ou des parties optionnelles.

Tâches de transformations

Les techniques de tracé permettent la création de formes simples comme décrit dans les tâches de saisie. La création et la manipulation de formes graphiques plus complexes utilise quelques techniques standards, notamment par l'utilisation de poignées de manipulation.

Poignées de manipulation

Pour modifier des attributs géométriques (position, taille, épaisseur, etc.), on utilise des *poignées*, petits rectangles pouvant être déplacés directement par un cliquer-tirer. Selon le cas, les poignées permettent de déplacer, changer la taille, transformer géométriquement, et de manière générale modifier la géométrie d'un objet.

Exemples :

- Dans un éditeur de schémas, la taille de chaque forme peut être changée en déplaçant ses poignées (4 ou 8).
- Dans un éditeur de schémas, les poignées permettent également de modifier les sommets d'un polygone et les points de contrôle d'une courbe.
- Dans un tableur, des poignées permettent de régler la largeur de chaque colonne et la hauteur de chaque ligne.
- Dans un traitement de texte des poignées permettent de régler les marges et la position des tabulations.
- Dans une boîte de dialogue ou une boîte de propriétés d'un éditeur de schémas, des poignées permettent de régler la taille des pointillés.
- Dans une boîte de dialogue d'impression ou de numérisation, des poignées permettent de définir la zone à imprimer ou à numériser.
- etc.

Modes d'interaction

Dans une interface graphique, un mode est un état de l'interface dans lequel les actions de l'utilisateur sont interprétées par le système de manière homogène et différente des autres modes.

On distingue deux types de modes : les modes spatiaux et les modes temporels. Un cas particulier de modes temporels sont les micro-modes. Comme on va le voir, les modes ne sont pas forcément une mauvaise chose, comme on le dit souvent. En réalité, ils sont indispensables, mais on peut les rendre "invisibles" ou presque grâce à quelques "astuces".

Modes spatiaux

Avec les modes spatiaux, l'interprétation des actions de l'utilisateur diffère en fonction de la position du périphérique de pointage. Par exemple, un clic dans une barre de menus provoque un effet différent d'un clic sur la barre de titre d'une fenêtre, sur un objet graphique, etc.

Ces modes passent inaperçus car il y a une mise en évidence graphique du lieu où ils se manifestent. C'est même leur disparition qui est souvent remarquée : lorsqu'une boîte modale est affichée, tout l'écran sauf la boîte de dialogue est dans le même mode spatial : on a l'impression que l'interface ne répond plus. Idéalement, il faudrait donner un feed-back de l'absence de mode spatial, par exemple en grisant l'ensemble de l'écran hors de la boîte de dialogue. C'est d'ailleurs cette technique qui est utilisée pour mettre en évidence les commandes inactives d'un menu : les commandes grisées ont un mode spatial différent des autres.

Modes temporels

C'est la notion la plus courante de mode : pendant un intervalle de temps (de l'activation du mode à sa désactivation), les actions de l'utilisateur sont interprétées de manière particulière. L'exemple typique est le mode insertion et le mode commande de l'éditeur *vi*. Comme il n'y a pas d'indication à l'écran du mode courant, l'utilisateur ne peut pas savoir comment vont être interprétées ses actions sauf s'il mémorise le mode courant.

Un exemple courant de mode temporel dans une interface graphique est l'utilisation d'une palette d'outils, comme dans MacPaint, MacDraw, etc. En fonction de l'outil sélectionné (par exemple l'outil de sélection, de création de rectangle, etc.) les mêmes actions ont des effets différents. Comme la palette n'est pas le point principal d'attention de l'utilisateur, même l'affichage du mode courant dans la palette ne suffit pas à éviter les erreurs. Une meilleure technique consiste à utiliser la forme du curseur (qui est le principal point d'attention) pour donner un feed-back du mode courant.

Les modes temporels sont en général activés par l'utilisateur. Ils sont parfois désactivés par le système lui-même (retour à un mode par défaut). Par exemple, dans MacDraw, après une action de création (mode création activé par sélection de l'outil Rectangle dans la palette par exemple), le système revient au mode sélection par défaut. Si l'on veut créer plusieurs rectangles à la suite, on doit resélectionner l'outil de création de rectangle à chaque fois. De manière générale, les changements de mode doivent être à l'initiative de l'utilisateur.

Les modes temporels sont couplés avec les modes spatiaux. Lorsqu'une boîte modale apparaît, le mode spatial qu'elle définit est actif pendant que le mode temporel l'est.

Micro-modes

Les micro-modes sont des modes temporels liés à une action physique continue de l'utilisateur. Par exemple, pendant un "drag" (déplacement de la souris avec le bouton enfoncé), on est dans un micro-mode : les actions de l'utilisateur (déplacement) sont interprétées différemment que lorsque le bouton est relâché.

Ces modes passent inaperçus car ils correspondent à des "modes" naturels dans le monde réel : pour prendre un objet et le déplacer, il faut le tenir. Cela explique pourquoi les menus déroulants du Macintosh sont plus faciles à appréhender que ceux de Windows : sur le

Macintosh, la sélection dans un menu utilise un micro-mode, tandis que dans Windows, c'est un mode temporel sans autre feed-back que la présence du menu à l'écran.

En résumé, il faut éviter les modes temporels et plutôt utiliser des modes spatiaux et des micro-modes. Dans tous les cas, il faut un feed-back adéquat pour informer l'utilisateur du mode courant.

Les modes sont une manifestation "locale" du dialogue homme-machine ou de la syntaxe de l'interaction. Si l'on choisit une syntaxe de type "verbe-objet", on impose un mode temporel par verbe, ou au moins un mode temporel de sélection de verbe et un mode temporel de sélection d'objet. Par contre si l'on choisit une syntaxe de type "objet-verbe", on utilise uniquement des modes spatiaux : la notion de sélection permet de définir l'objet de la commande par un mode de sélection spatial (sélection directe des objets) ; le verbe est sélectionné dans une palette ou un menu, qui sont comme on l'a vu des modes spatiaux. De plus, la connaissance de l'objet au moment de la sélection du verbe permet de ne rendre activable (mode spatial) que les commandes concernées. Cependant, cette syntaxe "objet-verbe" est difficile à utiliser pour les opérations de création puisque par définition ces opérations ne s'appliquent pas à des objets.

Les boîtes de dialogue modales sont un autre exemple de dialogue. Ces boîtes de dialogue sont généralement affichées à la suite de la sélection d'une commande dans un menu. Par exemple, une boîte de dialogue d'un outil de dessin permet en général de changer les attributs graphiques du (des) objet(s) sélectionné(s). Cette boîte est souvent modale car on estime qu'il faut définir les attributs, puis les valider avant de voir les objets graphiques modifiés : on impose un mode temporel et spatial. Ce mode est cependant superflu : le changement de valeur d'attribut peut très bien prendre effet immédiatement (plus de mode temporel) ; les objets peuvent être édités pendant que la boîte est présente à l'écran. Il suffit de la mettre à jour lorsque la sélection change. Evidemment, tout cela nécessite plus d'efforts de conception et de programmation ...