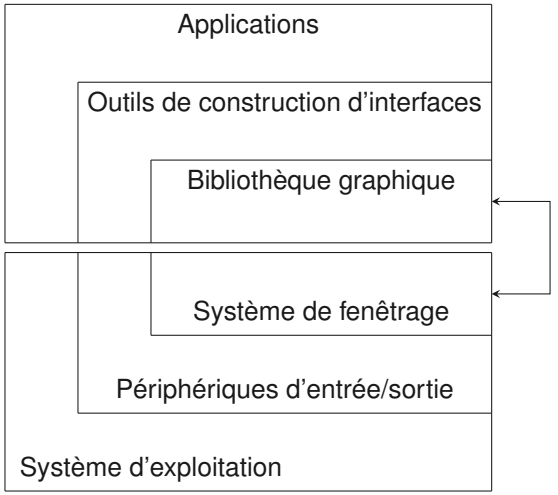
IHM: Fondements des Interfaces Graphiques

Olivier Chapuis
chapuis@lri.fr

Architecture générale des interfaces graphiques



Plan

Plan

Périphériques de sortie

Ecrans “bitmap” Cathodique, **LCD**, Plasma, OLED.

Taille exprimé en la dimenssion de la diagonal en pouce (1 pouce = 2.54 cm, 30 pouces $\sim$ 76cm) et le ratio largeur sur hauteur (e.g., 16/9).

Résolution exprimé en pixel (e.g., 2560x1600). La taille plus cette résolution donne une densité exprimé en général en “dpi”: dot[pixel] per[par] inch[pouce] (typiquement 100 dpi $\sim$ 40 pixels par cm, c.à.d. 1 pixel $\sim$ 0.25 mm).

Résolution en couleurs (“profondeur” RGB[A]): 8 bits (256 couleurs), 16 bits (65536 couleurs) ou 24[32] bits (16 millions de couleurs [+256 niveaux de “transluscence”])

Résolution temporelle exprimé en Hz qui est le nombre de trame [frame] par seconde que l'écran peut afficher (typiquement 60 Hz).

Exemple de bande passante:
60 img/s * 2560x1600 pixels * 3 octets/pixel $\sim$ 700 Mo/s

Périphériques de sortie







`videos/output/volumetric_coll.avi` `videos/output/volumetric.flv`
`videos/output/2004-VolumetricDisplay.mov`

Périphériques d'entrée

claviers, boîtes à boutons

potentiomètres (rotatifs, linéaires)

souris, tablettes, joystick, trackball

écrans tactiles, crayons optiques

reconnaissance de la parole

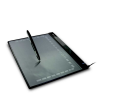
capteurs de position et de direction

vision par ordinateur

surfaces interactives

dispositifs hybrides entrée/sortie
Retour d'effort









videos/force/phantom.flv

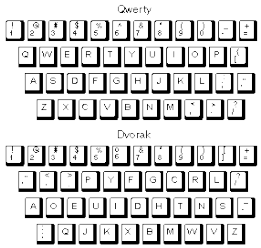
Entrée de texte: Le clavier

Un problème: Optimisation de la position des touches

Disposition Dvorak: 10 à 15% de gain de vitesse, réduction de la fatigue

Clavier Logiciel: optimisation $\Rightarrow$ pointage

Clavier avec des touches qui possèdent des écrans (oled) ou projection sur le clavier.



videos//output/keyboard-dpy.mov

Entrée de texte: alternative au clavier classique

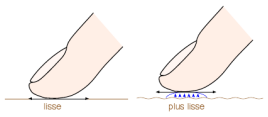
Clavier à accords (chord keyboards):
- Peu de touches (4 ou 5)
- Utilisation de plusieurs touches simultanément
- Saisie rapide avec une seule main

Téléphone portable:
- tape multiple
- tape plus ou moins long
- Système T9: un seul “tape” par lettre le système découvre le mot

Périphériques 3D, retour tactile et retour de force

Périphériques 3D ... bof, bof ...

Retour tactile: vibrations et surface plus ou moins dur. Peu d’exemples d’applications, vibrations lors que l’on passe sur certaines cibles, peut-on reproduire des vraies textures ?



Type de Contrôle et de Périphérique

Ordre 0: déplacement du périphérique d’entrée correspond à un déplacement d’un objet Exemple: Couple souris - curseur

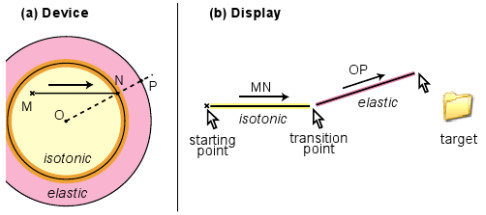
Ordre 1: le périphérique d’entrée contrôle la vitesse d’un objet
Exemples: couple (i) joystick - curseur; (ii) souris - dessin d’un vecteur vitesse

Périphérique isotonique: Contrôle de position – Ordre 0, débrayage pour les longues distances. Exemples: souris, touchpad, tablette

Périphérique élastique: Possède un état stable avec retour “élastique” à cet état – Ordre 1. Exemples: joystick (inclinaison ↔ vitesse)

Une périphérique isotonique et élastique

RubberEdge (Casiez et al. 2007): réduction du débrayage en combinant un contrôle positionnel et élastique



Centre du touchpad: position
Bords du touchpad: un système élastique pour contrôler la vitesse de déplacement

videos/pointing/rubber-edge.mov

Périphériques d'entrée: Control-Display Gain

Résolution: nombre d'impulsion que peut envoyer le periphérique d'entrée pour une distance donné. Souvent expriment en dpi = dot[impulsion] per[par] inch[pouce]. 1 pouce = 2.54 cm

Exemple pour une souris: entre 300dpi (lent), 600dpi (typique) et 2400dpi (max?), soit, respectivement une impulsion tous les 0.083, 0.042 et 0.01 mm

Controle-Display Gain [Gain contrôle-écran]:

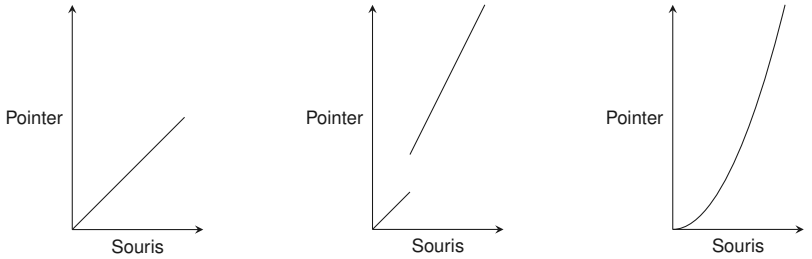
$$CDGain = \frac{\text{Distance parcourue à l'écran par le pointeur}}{\text{Distance parcourue par le périphérique d'entrée}}$$

Exemples. Ecran 100 dpi, souris 600 dpi et une impulsion souris $\leftrightarrow$ un pixel: $CDGain = 6$. Tablette de meme taille que l'écran avec contrôle direct tablette-écran: $CDGain = 1$.

Périphériques d'entrée: Accélération

Problème: Si le *CDGain* est trop grand il est difficile d'être précis, mais si il est trop petit on est obligé de débrayer la souris pour parcourir de "grande" distance.

Accélération: modification dynamique du *CDGain* en fonction de la vitesse de la souris. Plus on va vite, plus le *CDGain* devient grand.



CDGain Constant

X Window: Deux valeurs pour le *CDGain*

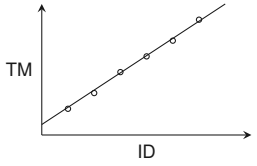
Mac OS X et Windows: Changement progressif du *CDGain*

Pointage: Loi de Fitts

$$TM = a + b.ID \quad \text{où}$$

$$ID = \log_2(\frac{D}{W} + 1)$$

MT: Temps de Mouvement
ID: Indice de Difficulté (de la tâche)
a, *b*: constantes empiriques (qui dépendent des utilisateurs, du peripherique ...etc.)

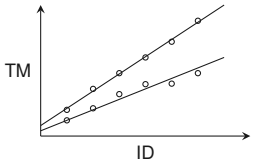


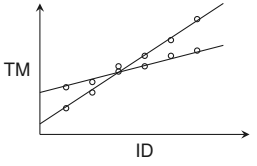
Loi de Fitts et IHM

Design d'interface (clavier logiciel, interface adapté à différent type d'utilisateurs, ...)

Technique de Pointage: reduire *D* ou agrandir *W*

Comparer les performances entre péripheriques d'entrée (souris, touchpad, tablette ...)



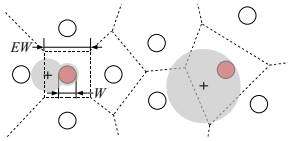


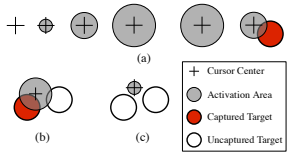
La pente de la droite comme un indice de performance

Techniques de Pointage

"Battre la Loi de Fitts" en réduisant *D* ou agrandissant *W*

- Prédiction de la position de la cible pour réduir *D*
- Agrandir la taille des cibles lorsque le curseur se trouve dessus (Mac OS X Doc)
  `http://profs.logti.etsmtl.ca/mmcguffin/research/expandingTargets/`
- Agrandir *W* dans l'espace moteur [Blanch et al.] demos/semantic-pointing/
- Bubble cursor [Grossman & Balakrishnan] videos/pointing/bubble-cursor.mov
- DynaSpot [Chapuis & al.] videos/pointing/CHI09-dynaspot.mov





Plan

Système de fenêtrage

Système de fenêtrage. Bas niveau du rendu (lien avec le matériel), partage des ressources (sorties vidéos, entrées clavier-souris) grâce au concept de fenêtre

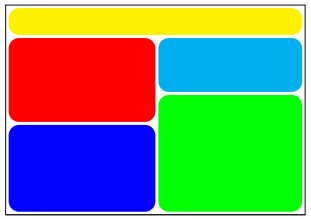
Fenêtre. Une surface (rectangulaire) d'affichage utilisée par une application pour réaliser les entrées/sorties. Les fenêtres permettent de partager la sortie vidéo entre les applications.

Gestionnaire de fenêtres. Haut niveau, placement et ordre en Z des fenêtres (superposition des fenêtres), décorations des fenêtres, interactions avec l'utilisateur (deplacement, redimensionnement, maximisation et iconification des fenêtres)

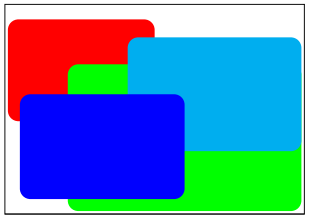
Utilitaires de Gestion de fenêtres. Barre de Tâche, Gestionnaire de bureaux virtuels, notifications ...etc.

Interface Utilisateur Graphique. L'intérieur des fenêtres, le niveau applicatif visible où se déroule l'interaction
Environnement de Bureau. Ensemble "cohérent" présenté à l'utilisateur (gestionnaire de fenêtres + utilitaires de gestion de fenêtres + ensemble d'applications)

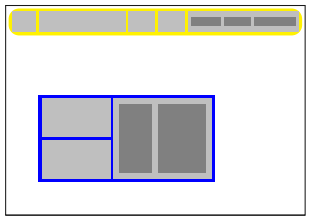
Modèles de fenêtrage



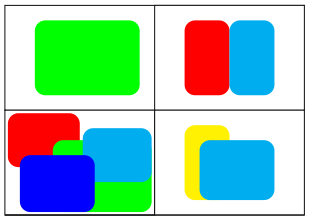
Pavage



Recouvrement

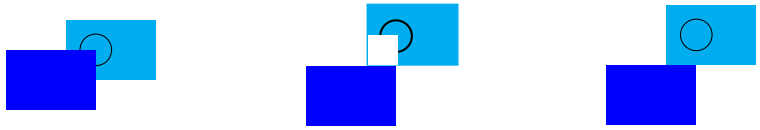


Arbre



4 écrans virtuels

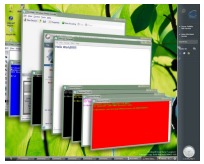
Modèles de fenêtrage: Dessin



Dessin sur l'écran: Réaffichage des parties cachées via un évènement du système de fenêtrage à l'application.

Mais, de plus en plus, on fait différement: on dessine "out of screen" fenêtre par fenêtre, puis on "compose" l'écran avec les fenêtres. Permet, par exemple, des effets et des fenêtres semi-transparentes.

Plus coûteux en mémoire, mais plus rapide et donne une interaction plus fluide ... ceci n'est plus un problème avec nos GPU acutels ...



Modèles de fenêtrage: Entrées

Gestion des entrées: le système passe les événements souris et clavier à une fenêtre (fenêtre qui à le focus).

Événements du système de fenêtrage aux applications: changement de focus, entrée-sortie de fenêtre

Politique de changement de focus:

- Clique pour le Focus – clique passé à l'application ou non (non avec Mac OS X)
- Le Focus suit la "souris" – possible avec X Window

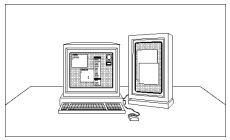
Y-a-t-il une politique meilleure que l'autre ?

Problème connexe: ordre en z des fenêtres.

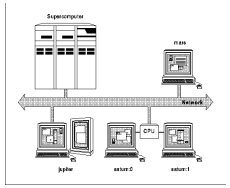
videos/wm/CHI07-CopyPaste.mov

Le Système X Window

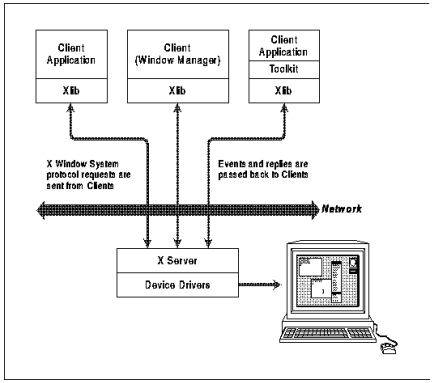
X is a network-oriented windowing system with a Server-Client Model



Display = Keyboard + mouse + screenS



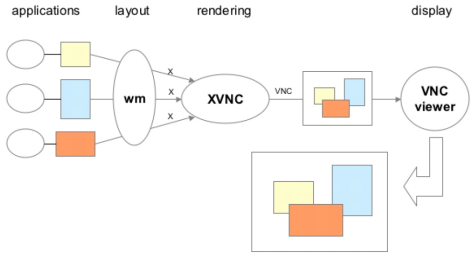
Applications can run across the network



X Window System architecture

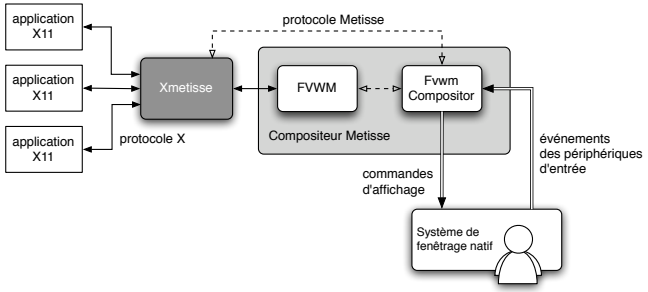
X is extensible (XInput, XRender, XRandr, XComposite, XDamage, MPX ...)

Remote Desktop / VNC (Virtual Network Computing)



Xvnc renders the "display" in memory, making the desktop image available to vnc viewers (via the net). The vnc viewers forwards mouse and keyboard events to the Xvnc server.

Metisse



Xmetisse: a X server which does off-screen rendering | per window |

FVWM + **FvwmCompositor**: On-screen rendering (compositing) in OpenGL (window = textured polygone), Window management and User interaction

videos/wm/UIST05-metisse.mov / videos/wm/UIST06-facades.mov

Gestion des Fenêtres: Problématiques

Les activités se multiplient traitement de texte, tableur, jeux, courrier électronique, Web, messagerie instantanée, édition d'images, de sons et de vidéos, media players etc.

De plus en plus de fenêtres !

Exemples de Problèmes de recherches:

- Placement des fenêtres
  videos/wm/2000-dynamic-space.mp4 videos/wm/1996-elastic-windows.mov
- Pavage, recouvrement et tab
  videos/wm/2001-overlapping-windows.mov
- Passer d'une fenêtre à l'autre "Window Switching Problem" et correspondance tâche $\leftrightarrow$ ensemble de fenêtres
  videos/wm/2004-ScalableFabric.mov videos/wm/2000-task-gallery.mpg

Plan

Librairies graphiques

Modèle de dessin

- Dessin direct (modèle du peintre)
- Dessin structuré : description structuré (e.g., via avec du xml)

Définition d'objets graphiques

- primitives graphiques lignes, courbes (splines, bezier), Polygones (triangles, rectangles, etc.), arcs, cercles, ellipses, textes, images bitmap, ... etc.
- Géométrie: taille et position
- Attributs graphiques: couleur, épaisseur des lignes, fonte, texture, gradient, transparence, éclairage ... etc.

Librairies/Modèles graphiques

- Dessin direct: Xlib (X Window), Java2D, OpenGL
- Dessin structuré: PDF, SVG, Cario

Librairies graphiques: Autres Fonctions

Clipping (dessin limité à une partie de l'écran) par masque bitmap ou région rectiligne (horizontales et verticales uniquement), polygonale, définie par des lignes et courbes (ex : PostScript)

Anti-aliasing, gestion de la transparence

Conversions entre espaces de couleur (RGB, HSV, CMY, etc.)

Structures de données pour l'affichage (graphe de scène)

Librairies graphiques: Systèmes de coordonnées

Coordonnées liées au périphérique

- unité = pixel
- coordonnées écran ou fenêtre ? quelle orientation et origine ?

Coordonnées physiques

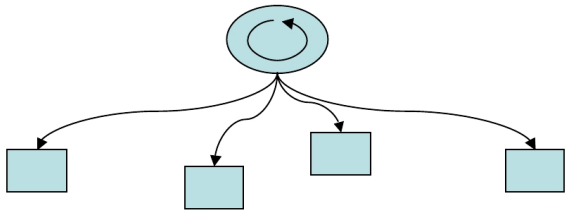
- unité = cm, pouces, etc.
- système indépendant du périphérique mais lié à la technique de présentation utilisée

Coordonnées du modèle

- unité = m, km, etc.
- Indépendant de l'application

Librairies graphiques: Événements

```
tantque non fini faire
    attendre jusqu'à file non vide      // attente passive
    ev := tête de file                  // extraire événement
    cible := chercherCible(ev)
    si cible ≠ NIL alors cible.traiter(ev)
fin tantque
```



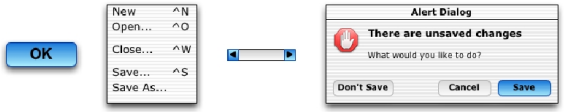
Style très différent de la programmation algorithmique

Outils de construction d'interfaces (boîtes à outils)

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

Abstraction atomique: le widget

- Objet interactif, Composant logiciel
- Bouton, menu, barre de défilement, boîte de dialogue, ...



Un widget = trois facettes

Présentation – Comportement – Interface d'application

Application = arbre de widgets

- Noeuds : conteneurs (barre de menus, boîte de dialogue, ...)
- Feuilles : widgets simples (boutons, barres de défilement, ...)

Outils de construction d'interfaces (boîtes à outils)

Il existe de nombreuses boîtes à outils

- Xt, Motif, GTK (C) sous Linux
- MFC (C++) et ses dérivés sous Window
- Carbon (C), 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

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 Widgets

Exemple : les widgets de base du Macintosh (1984)

- bouton (button) et label
- 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)

D'autres exemples:

- Barre de Progression
- Bulle d'aide (Tooltips)
- Arbre d'Icones
- Haut Niveau: Sélecteur de Fonte et de Couleur, Calendar

`http://www.digitalfanatics.org/e8johan/projects/widgets/`

`http://library.gnome.org/devel/gtk/unstable/ch02.html`

Exemple de service : Placement des widgets

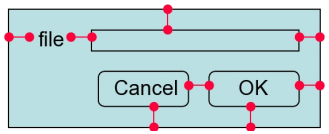
Interface/Application = arbre de widgets $\Rightarrow$ Placement des widgets

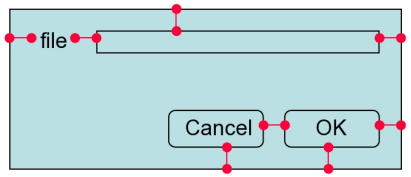
Règles générales

- Imbrication géométrique d'un widget fils dans son parent
- Contrôle par le parent du placement de ses fils

Algorithme de placement

- Taille naturelle de chaque fils
- Taille et position finales imposées par le parent
- Contraintes : Grille, formulaire, etc.
- Placement dynamique





`videos/toolkit/artistic-resizing.mov`

Facettes d'un widget

Présentation

- Apparence graphique
- Plus ou moins Paramétrable (e.g. Look and Feel Java)

Comportement

- Réaction aux actions de l'utilisateur
- Peu ou pas paramétrable

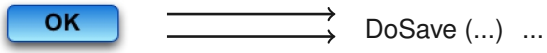
Interface d'application (Liens widgets - application)

- fonctions de rappel (callbacks)
- valeurs actives
- envoi de message

Interface d'application : fonctions de rappel (callbacks)

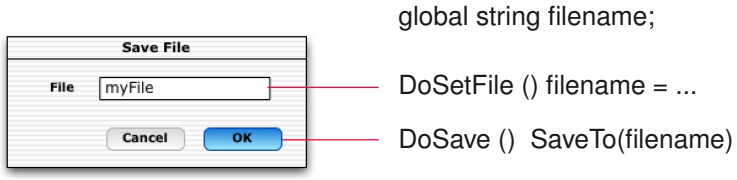
1. Enregistrement lors de la création du widget

2. Appel lors de l'activation du widget



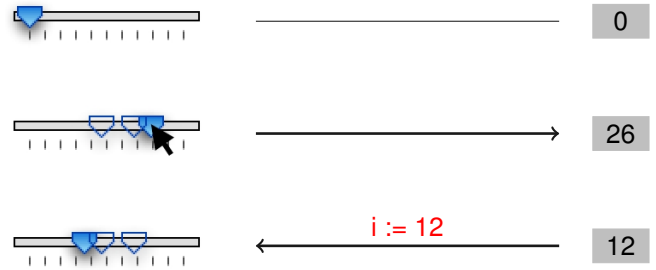
Problème : "spaghetti" des callbacks

Partage d'état entre plusieurs call-backs par variables globales



Interface d'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



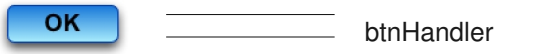
Problèmes: Limité aux types simples (e.g., entier), Lien de retour peut être coûteux

Avantages: Vues multiples

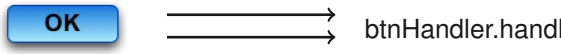
Interface d'application : envoi de message

Association d'un objet à un widget et de méthodes de l'objet aux changements d'état du widget

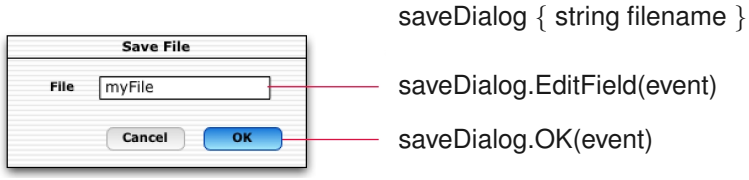
1. Un objet est associé au widget au moment de sa création



2. Des méthodes de cet objet seront appelées par le widget



Avantage : permet une bonne encapsulation des données



Exemple de service : squelettes d'application

Principe : "Application à trous"

- Code d'une application incomplète, à compléter
- Contient ce que l'on ne peut pas mettre dans une toolkit
  - Structure globale de l'application
  - Fonctionnalités globales (historique, copier-coller, ...)
  - Interactions "non widgets" (drag-and-drop)
- Trahit l'inadéquation des langages de programmation

Exemple : MacApp (Apple, 1986)

- Notion de document (contenu d'une fenêtre)
- Notion d'action (que l'on peut faire, défaire)

Générateurs d'interface

But : aider (automatiser ?) la mise en oeuvre d'interfaces

- placer les widgets, modifier leurs attributs (couleur, etc.)
- les connecter à l'application
- tester leur comportement

De nombreux noms pour une même famille de systèmes

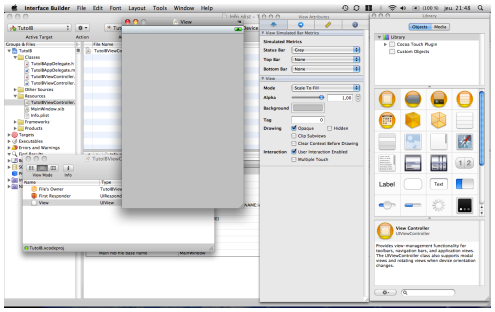
- user interface management systems (UIMS)
- user interface builder
- user interface development environment

Exemples : Visual* sous Windows, Interface Builder sous OS X

Générateurs d'interface

Inconvénients : What You See Is All You Get

- solutions partielles
- il reste tout de même du code à écrire...
- le code généré est souvent difficile à modifier



Interface Builder: `http://www.youtube.com/watch?v=vixD-N6cbDY`

MaggLite: `videos/toolkit/MaggLitePuzzle.avi`

`http://www.emn.fr/x-info/magglite`

A LIRE

doc/notes-cours2.pdf

doc/notes-cours3.pdf

et toujours: doc/dui-chapter1.pdf