Université du Québec
École de technologie supérieure

ELE 615
GRAPHISME ET INTERFACE USAGER

ARCHITECTURE LOGICIELLE DES INTERFACES GRAPHIQUES (GUI)

Eric Fimbel
Département de génie électrique

Dernière révision 2006-01-17 16:00

Origine des outils et techniques

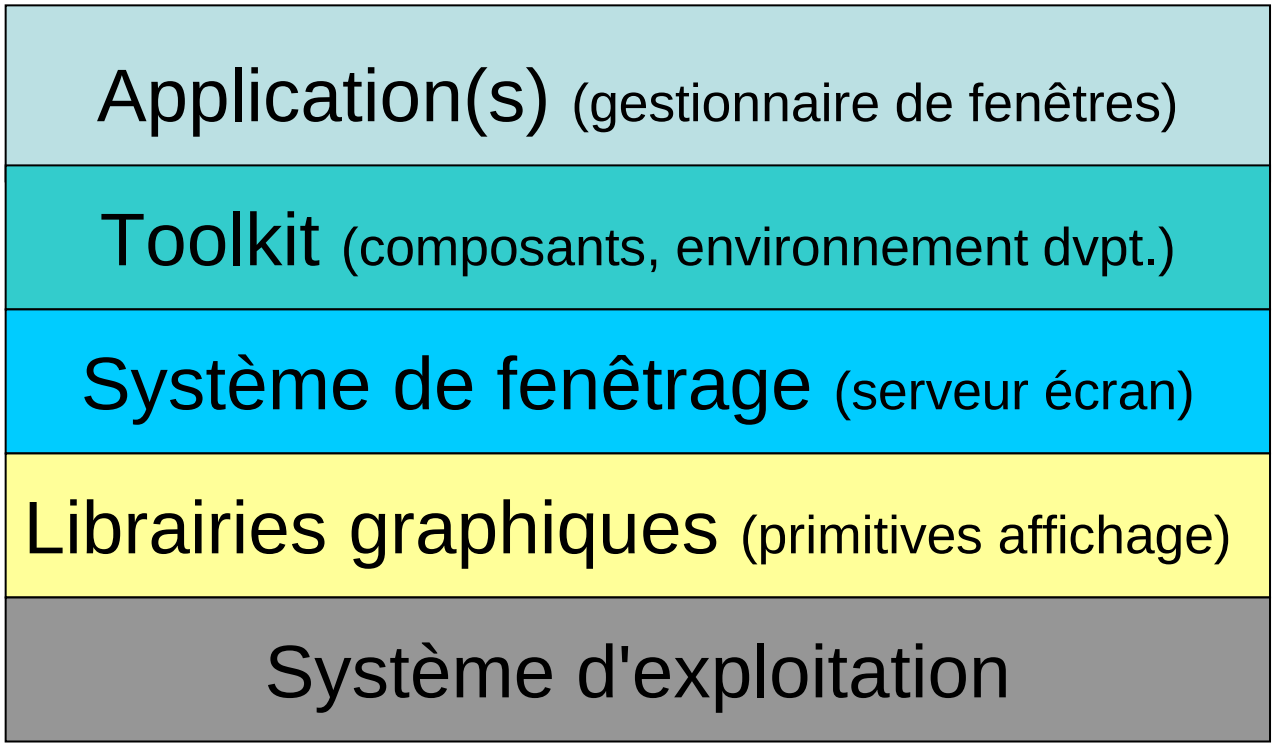
- Centre de recherche Xerox (PARC) , années 70.
  - Premiers widgets, Langages objets.
- Projet Athena, MIT (1985).
  - Architecture client-serveur.
- Microsoft VB (1991)
  - Programmation visuelle.
- Etc.
- Références
  - Myers, B., Hudson, S. E., & Pausch, R. (2000). Past, Present and Future of User Interface Software Tools. ACM Trans Comp. Hum. Interaction, 7(1), 3-28.

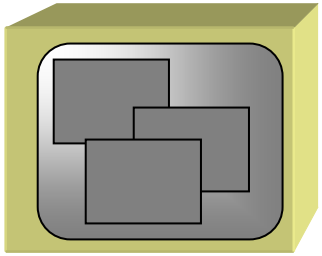
Outils et techniques actuels

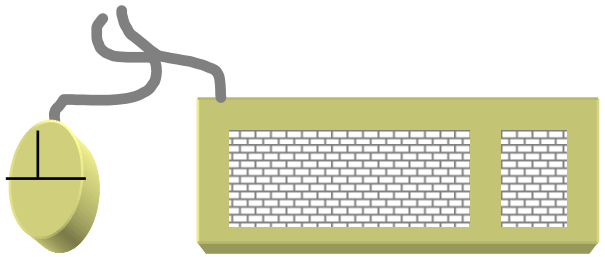
1. Gestionnaires de fenêtres et toolkits de développement
   - Toolkit : composants + environnement d'exécution / développement
2. Librairies graphiques.
   - Open GL, Java3D, Maya, etc. voir http://dmoz.org/Computers/Programming/Graphics/Libraries/
3. Langages orientés objet et par événements
   - Et langages de programmation dédiés (ex. Flash(R))
4. Langages de scripts de haut niveau
   - Python, Perl
5. Outils de programmation visuelle.
   - Ex. Visual Basic
6. Composants d'application
   - "Application widgets" , éditeur, fureteur...
   - Ex. Microsoft OLE, Java Beans.

Architecture logicielle idéale

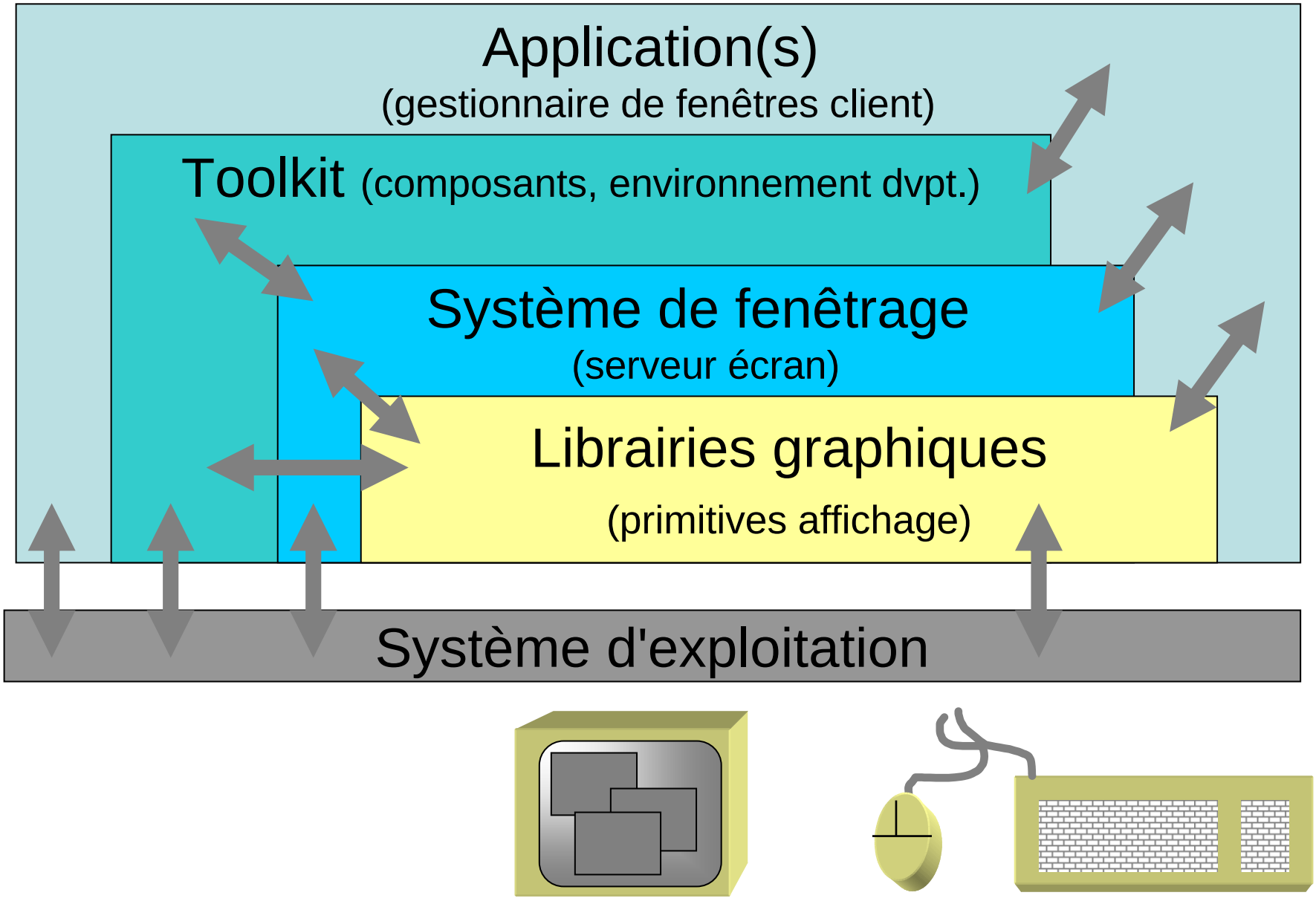
- Couches logicielles durant l'exécution.







Architecture logicielle réelle

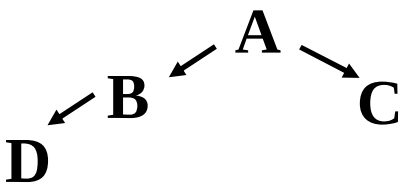


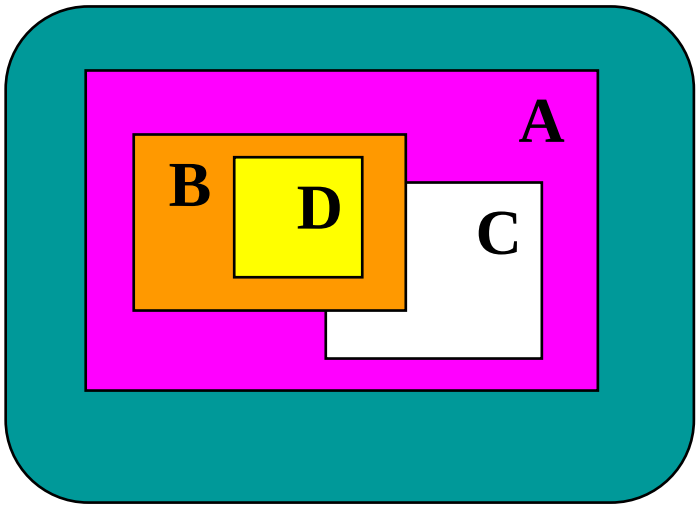
Système de fenêtrage

- Autres noms
  - "Serveur écran", "window server", "display server", "x-server", "windowing system".
- Contrôle l’accès aux ressources système par l'application
  - Régions de la mémoire vidéo, etc.
- Transmet les entrées à l'application.
  - Clavier, souris,...Programmation visuelle.
- Exécute les demandes des applications.
  - Demandes bas niveau, concernant écran et ressources.
  - Affichage de régions, redirection des événements...
- Rendu graphique
  - Par des appels à la librairie graphique (ou codé en interne).

Fenêtrage

- Fenêtre = région d'écran
  - Avec quelques décorations: boutons maximiser, minimiser, titre....
- Organisation hiérarchique
  - Fenêres et sous-fenêtres, autres sous composants
- En théorie, l'écran est une fenêtre.
  - Pas toujours accessible depuis l'application



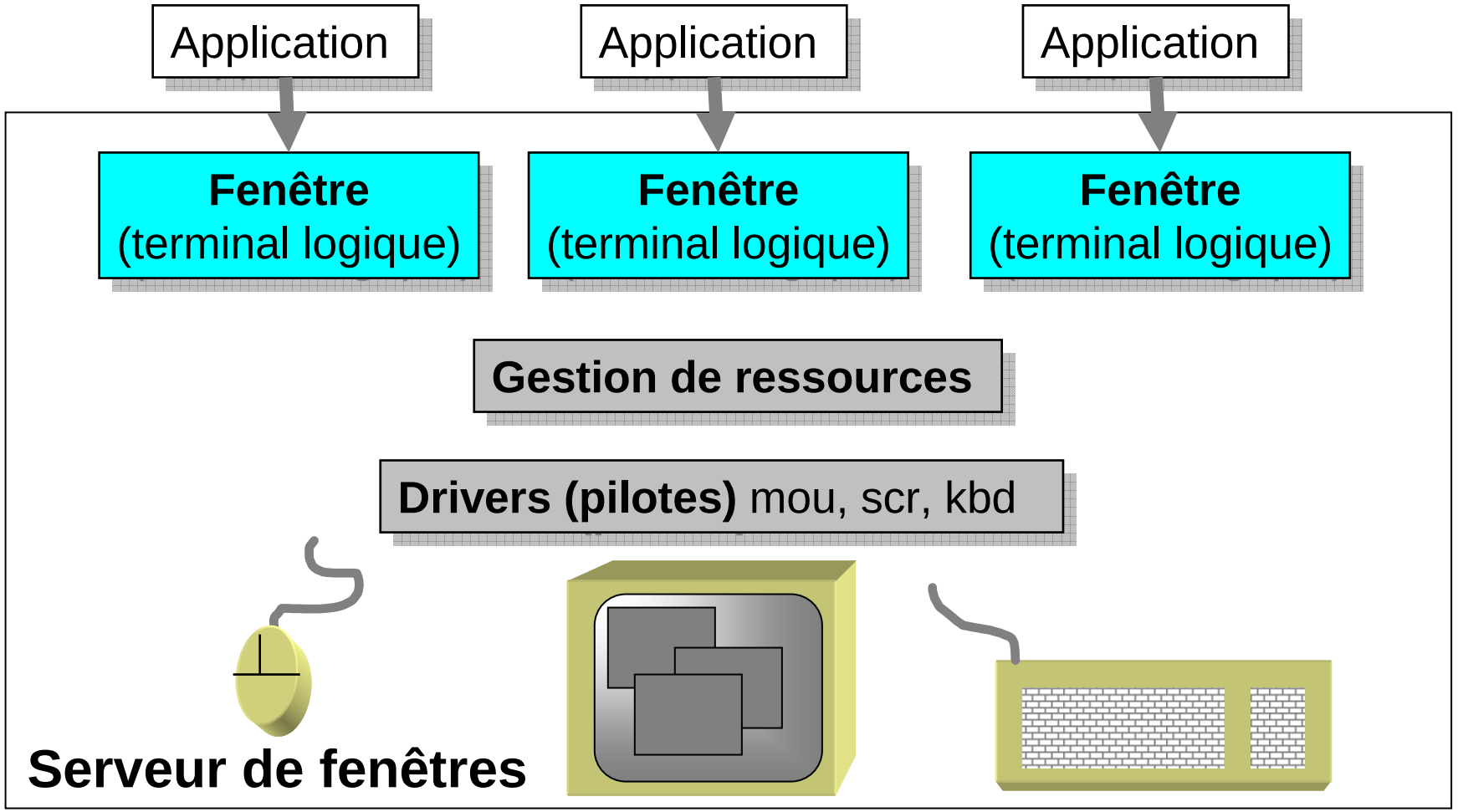


Gestionnaire de fenêtres client

- Autres noms
  - "Windows manager",
- Gère les fenêtres principales
  - Ouverture, destruction, chevauchement...
  - Relations entre fenêtres principales
- Gère l'apparence et l'organisation interne des fenêtres
  - Maximisation, redimensionnement...
  - Interne aux fenêtres principales
- Gère le style d'interaction
  - Activation ou désactivation clavier, drag-and-drop...

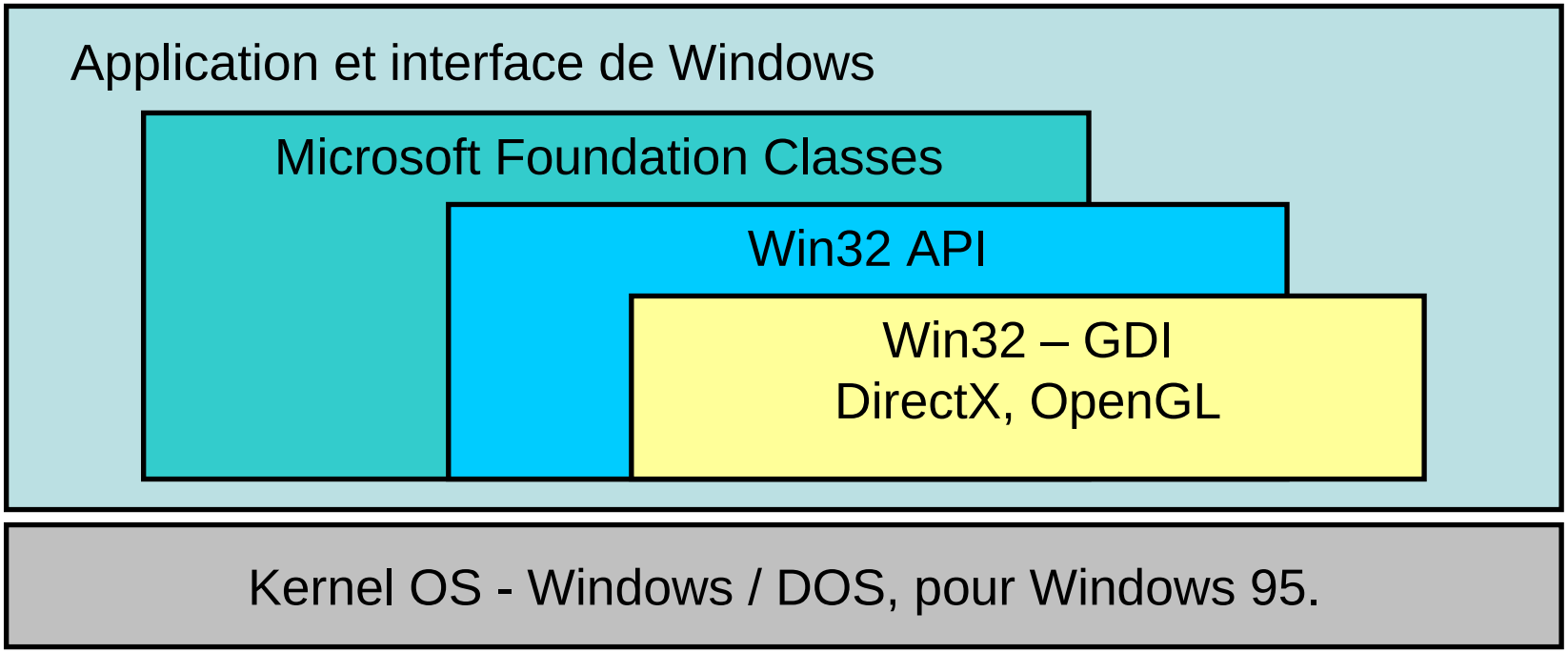
Architecture d'interface client serveur

- Architecture généralement adoptée dans les toolkits / OS.
  - Système de fenêtrage = serveur de fenêtres
  - Gestionnaire de fenêtres et applications = clients.



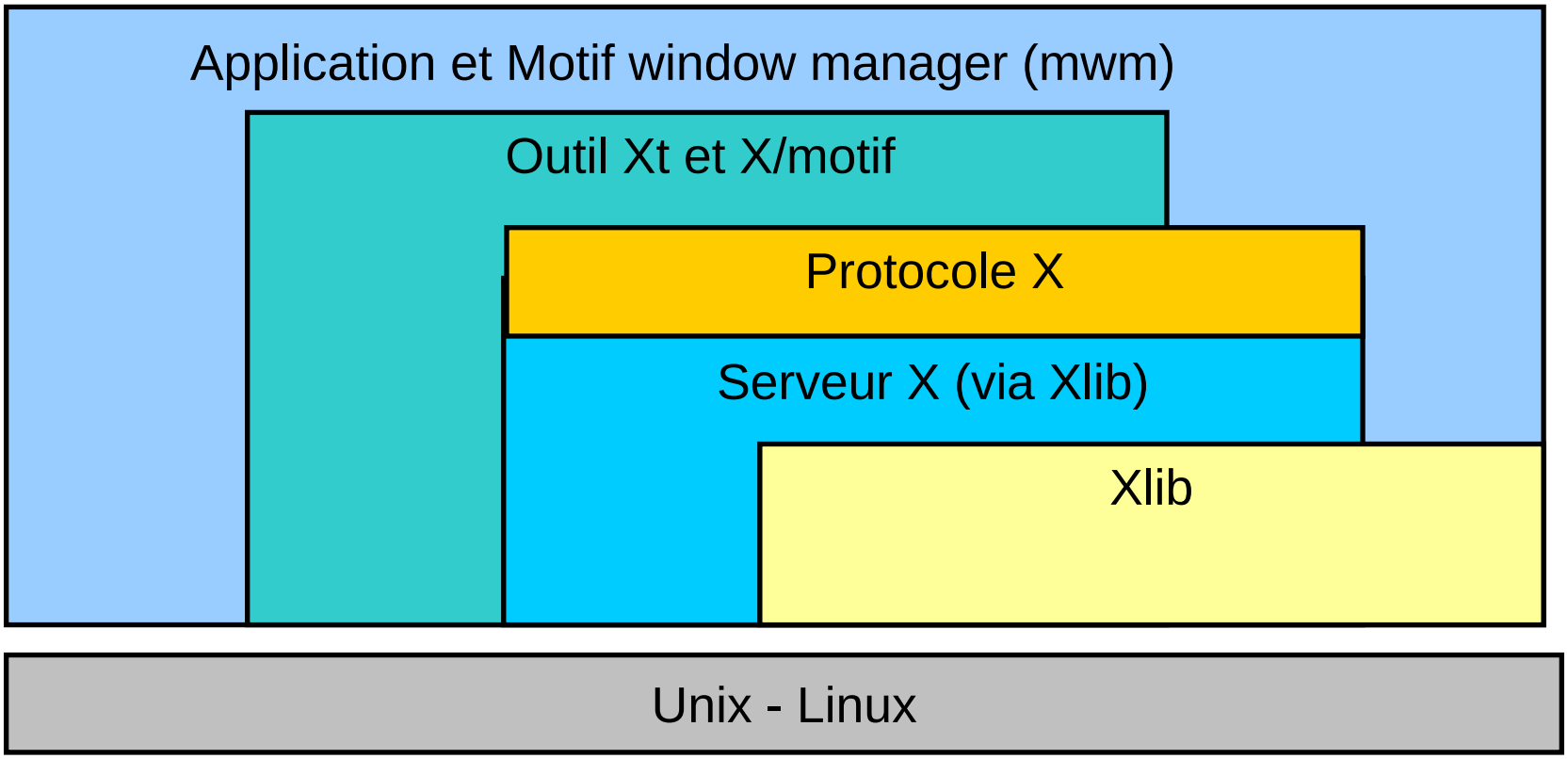
Architecture client-serveur : PCs

- Ordinateurs personnels: tout roule sur le même processeur
  - Librairie, toolkit, gestionnaire d'écran inclus plus ou moins dans l'OS
  - Exemple: Mac OS.
- Architecture logicielle pas vraiment en couches
  - Exemple: Windows

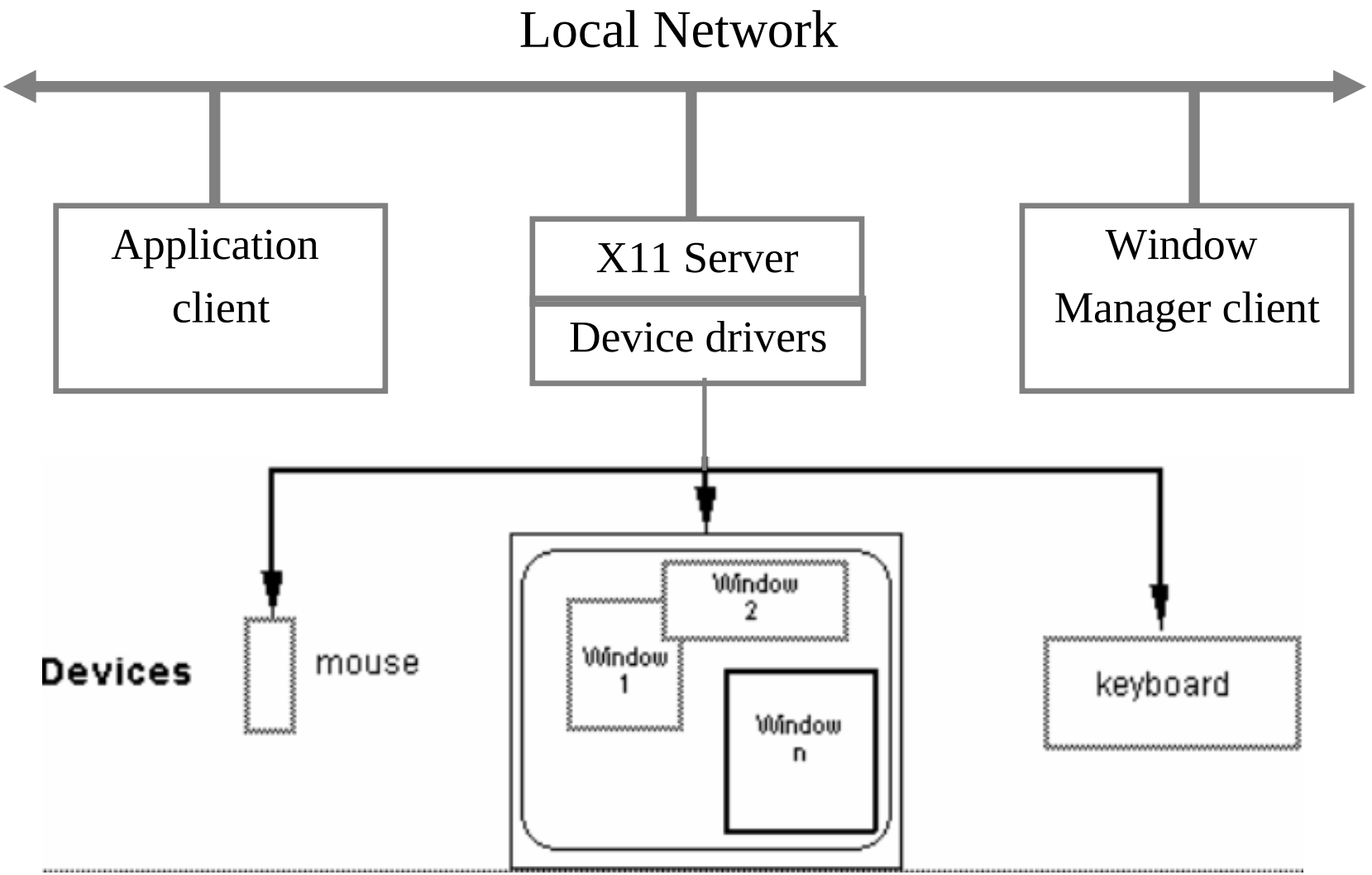


Architecture client-serveur: X-windows

- Réseau d'ordinateurs
  - Clients = applications, serveurs = gèrent les ressources, écrans, imprimantes, etc.
  - Portable et flexible



Architecture client-serveur: X-windows

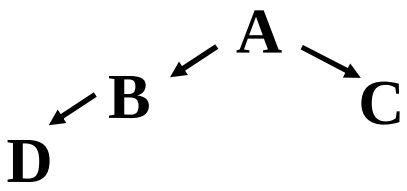


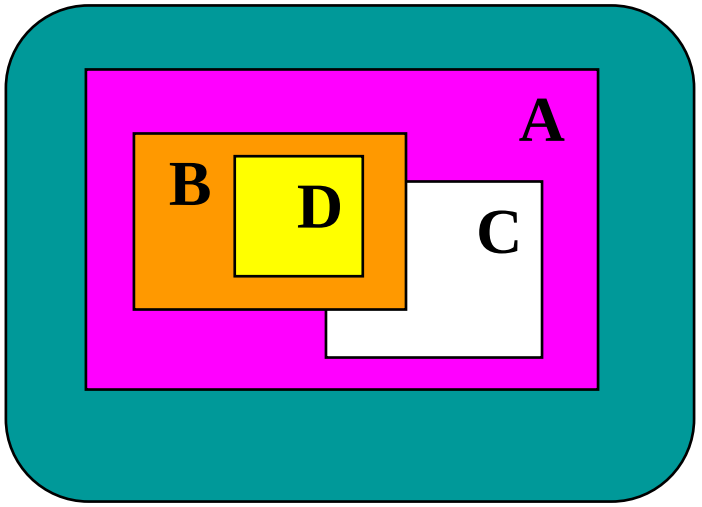
Programmation par événements

1. Les pilotes engendrent des événements.
   - Souris: « se déplace en X,Y », « bouton clické ».
   - Clavier: « touche pressée », « touche relâchée ».
2. Le système de fenêtrage (serveur écran) traite les événements.
   - Les complète et les met en forme (« code ascii A envoyé »).
   - Engendre des événements de haut niveau (focus perdu, gagné, caractère tapé...).
3. Le système de fenêtrage distribue les événements aux applications.
   - En général, le client est l'application qui a le **focus** ("active").
4. Les applications répondent aux événements
5. Les applications peuvent aussi engendrer des événements.

À qui va un événement de souris

- Règle hiérarchique
  - Composant le plus spécifique, son parent (conteneur), etc.
  - ...Jusqu'à ce que l'événement soit traité.
- Ordre de placement
  - En cas de conflit, prendre les fenêtres de l'avant vers le fond



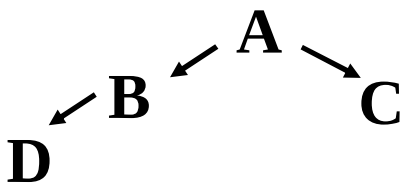


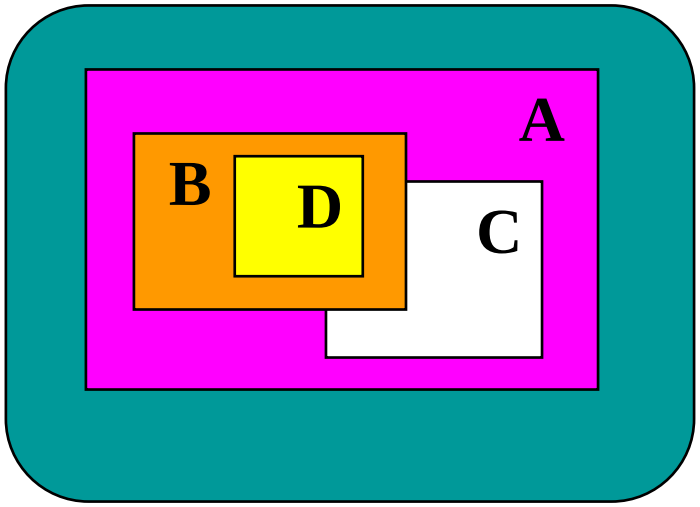
À qui va un événement de clavier

- Stratégie 1: composant qui a le focus
  - Puis son parent, etc jusqu'à ce que l'événement soit traité.
  - Focus: explicite (clic obligatoire) ou implicite (sous la souris)

- Stratégie 2: capture par un composant
  - Interaction modale.
  - Exemple fenêtre de dialogue.

- Roulette de souris:
  - Se comporte comme clavier.





Événements: boucle de lecture-traitement

- Initialement sur les Mac Os.
- 1 thread pour l'application active.
  - Thread = file d'exécution. Événements mis et traités en ordre.
- L'application lit un événement et le traite.
  - Read (my-event)
  - If my-event = xxx ...

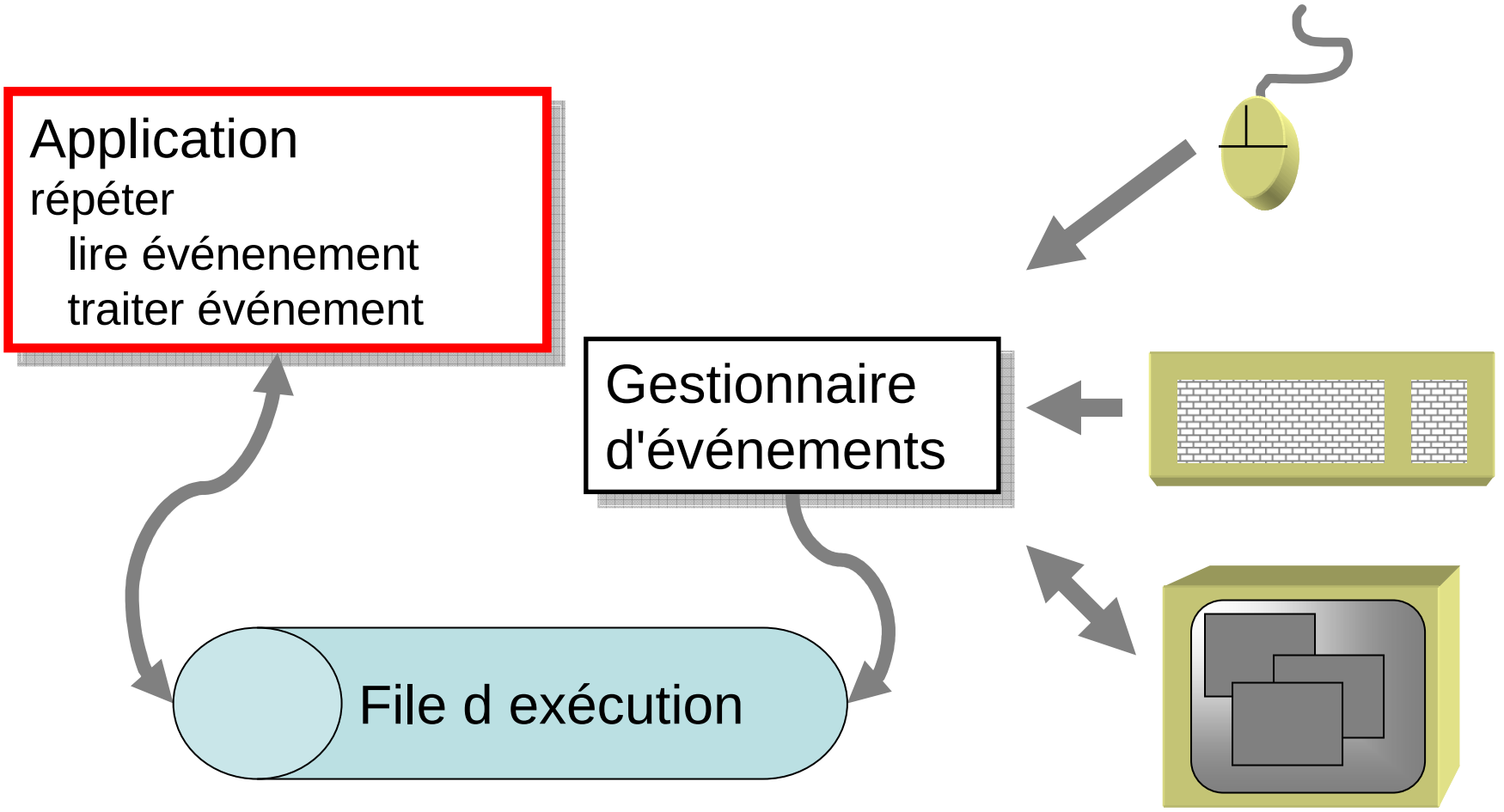
➢ Avantages.

- Simple à programmer, bon contrôle des interactions entre applications.

➢ Inconvénients.

- Il faut programmer tous les cas,y compris les événements traités par d'autres applications.

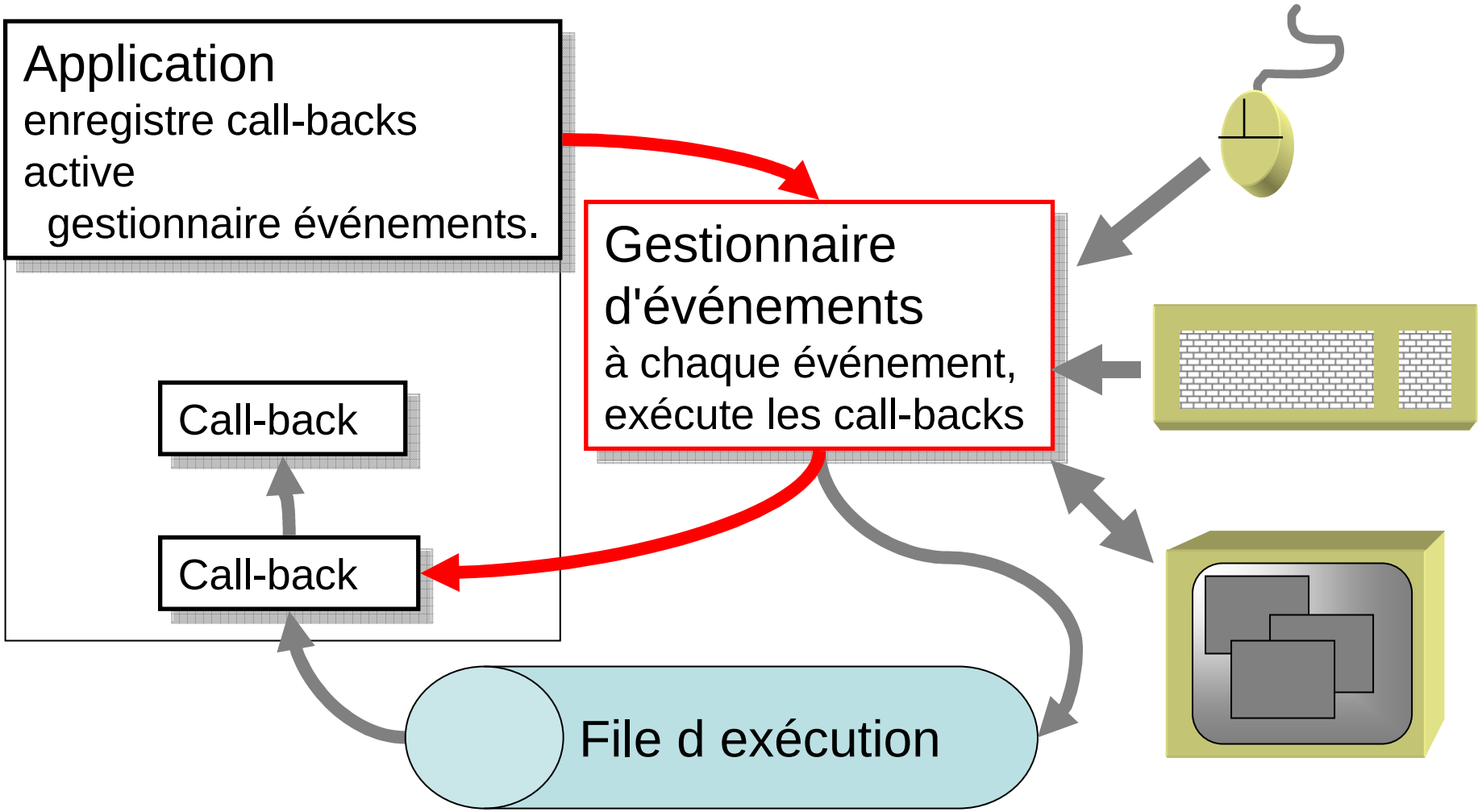
Événements: boucle de lecture-traitement



Événements : version basée notification

- Aussi appelée par "call back"
  - Exemple: Java Swing.
- 2 ou 3 threads (files d’exécution). Par priorité décroissante :
  1. Gestionnaire d’événements
  2. Application .
  3. Thread d'exécution des commandes d'affichage (parfois).
- Responsabilités de l’application.
  - Enregistre ses call-backs (« listeners », pilotes d’événements).
  - Active le gestionnaire d’événements.
  - Le gestionnaire d'événement connaît les call-backs à exécuter pour chaque type d'événements.
- Le gestionnaire d’événements fait deux choses.
  - Reçoit les événements.
  - Active immédiatement les call-backs associés à chaque événement.

Événements : version basée notification



Événements : version basée objets

- Utilise la hiérarchie d'objets**.**
  - Exemple: Java Swing, QT.
- Une fois qu'un composant a un événement, il est transmis selon les règles d'héritage
  - Un événement va être envoyé à PLUSIEURS composants hiérarchisés ou parallèles
- Exemple: Java 1.3
  - Événements clavier envoyés à TOUS les sous-composants du frame,
  - Seul le champ éditable qui a le focus l'affiche...
  - Mais les autres composants peuvent réagir.
  - Devenu faux en 1.4

Programmation par événements

- Exemple. Journal d'événements (LOG).
  - Action = touche 'a', mouvement de souris de bas en haut et clic.

```
java.awt.event.KeyEvent   KEY_PRESSED keyCode=65 on frame0
java.awt.event.KeyEvent   KEY_TYPED keyChar='a'on frame0
java.awt.event.KeyEvent   KEY_RELEASED keyCode=65 on frame0
java.awt.event.MouseEvent MOUSE_MOVED (462, 495) on frame0
...
java.awt.event.MouseEvent MOUSE_MOVED(340, 351) on frame0
java.awt.event.MouseEvent MOUSE_ENTERED(338, 349) on frame0
java.awt.event.MouseEvent MOUSE_ENTERED(84, 29) on javax.swing.JButton
java.awt.event.MouseEvent MOUSE_MOVED(84, 29) on javax.swing.JButton
...
java.awt.event.MouseEvent MOUSE_MOVED(70, 13) on javax.swing.JButton
java.awt.event.MouseEvent MOUSE_PRESSED(70, 13) on javax.swing.JButton
java.awt.event.FocusEvent FOCUS_LOST on javax.swing.JTextField
java.awt.event.FocusEvent FOCUS_GAINED  on javax.swing.JButton
java.awt.event.MouseEvent MOUSE_RELEASED(70, 13) on javax.swing.JButton
java.awt.event.MouseEvent MOUSE_CLICKED(70, 13) on javax.swing.JButton
```

Résumé et remarques finales

1. Architectures logicielles et outils de développement datant des années 70-90.
2. Gestionnaire de fenêtre, toolkit, gestionnaire d'écran, librairie.
3. Architecture en couche plus ou moins respectée
4. Modèle client - serveur plus ou moins respecté
5. Programmation orientée objet et par événements.
6. Architecture Modèle-Vue-Contrôleur (parfois)