Définitions et concepts clés

M322 — Concevoir et implémenter des interfaces utilisateur

Jérôme Frosssard

EPAI

18 mai 2025

Notion de programme interactif

Un programme interactif est conçu pour permettre à un·e utilisateur·rice d'interagir directement avec lui à l'aide de périphériques d'entrée-sortie.

Par contraste, un service, un démon (*daemon*) ou une tâche planifiée s'exécute de manière autonome sans aucune interaction avec un utilisateur·rice.

On peut distinguer deux types de programmes interactifs :

- Programme interactif séquentiel : Le programme contrôle les interactions, l'utilisateur réagit aux sollicitations du programme.
- Programme interactif piloté par événement : L'utilisateur·rice contrôle les interactions, le programme réagit aux actions de l'utilisateur·rice.

Interface utilisateur·rice

Une interface utilisateur·rice (*user interface* ou UI) est la partie d'une application qui permet à un·e utilisateur·rice d'interagir avec les fonctionnalités de cette application par l'intermédiaire de divers périphériques d'entrée-sortie :

- Terminal ou émulateur de terminal.
- Écran, clavier, dispositif de pointage (souris, trackpad, écran tactile, etc.)
- Casque de réalité virtuelle, joystick, manette de jeux, manette 6DoF, etc.
- Périphériques d'accessibilité : lecteur d'écran, ligne braille, clavier et souris ergonomiques, commande oculaire, etc.

Il est important de noter qu'une interface utilisateur·rice est strictement logicielle. Une UI nécessite des périphériques d'entrée-sortie, mais ceux-ci n'en font pas partie.

Différents types d'UI

On peut distinguer principalement deux types d'UI :

- UI en ligne de commande (*command line interface* ou CLI) : l'utilisateur·rice et le programme interagissent uniquement avec du texte par l'intermédiaire d'un terminal ou un émulateur de terminal.
- UI graphique (*graphical user interface* ou GUI) : l'utilisateur·rice manipule directement des éléments visuels interactifs (zone de texte, bouton, menu, etc.) dessinés à l'écran à l'aide d'un dispositif de pointage et d'un clavier.

Il existe également d'autres types d'UI, notamment :

- Les UI contrôlées par la voix (*voice-controlled user interface* ou VUI) utilisé principalement dans les assistants vocaux (Siri, Alexa, Google Assistant, etc.)
- Les UI contrôlées par les gestes (*gesture-controlled user interface*) utilisées dans les jeux vidéo avec des contrôleurs 6DoF (contrôleur VR, Wiimote, etc.) et dans certaines applications industrielles ou médicales.

Types d'UI et types de programmes interactifs

Le type d'interface (CLI ou GUI) et le type de programme interactif (séquentiel ou piloté par événement) sont deux choses distinctes.

Même s'il est généralement plus courant :

- De réaliser un programme interactif séquentiel avec une CLI.
- De réaliser un programme interactif piloté par événement avec une GUI.

Toutes les combinaisons sont possibles et un même programme peut avoir plusieurs interfaces utilisateur·rice·s.

Interfaces utilisateur·rice·s graphiques (GUI)

Dans ce module, nous nous intéressons en particulier aux interfaces utilisateur·rice·s graphiques (GUI).

Une GUI s'inscrit le plus souvent au paradigme WIMP (Window, Icons, Menus, Pointing device) :

- Un programme dessine son interface dans une fenêtre, c.-à-d. une zone rectangulaire attribuée au programme par le système de fenêtrage du système d'exploitation.
- L'interface est composée d'éléments graphiques de base (zone de texte, bouton, menu, etc.) et d'éléments graphiques spécifique (formes géométriques, élément de diagramme, texte riche, etc.).
- L'utilisateur·rice manipule directement l'interface du programme à l'aide d'un dispositif de pointage (souris, écran tactile, etc.).

Système de fenêtrage

Un système de fenêtrage est une partie du système d'exploitation, souvent optionnel, dont les buts sont :

- Permettre à plusieurs applications d'utiliser la même surface d'affichage sans avoir à se soucier les unes des autres.
- Permettre l'utilisation de dispositif de pointage (souris, écran tactile) en plus du clavier.

Et dont les tâches sont :

- Attribuer à chaque application une surface d'affichage virtuelle dont la partie visible est la fenêtre principale.
- Gérer le déplacement et le redimensionnement des fenêtres.
- Diriger les données des périphériques d'entrée (clavier, souris, écran tactile …) vers la bonne application.

Fenêtre

Une fenêtre est une portion de l'affichage, généralement rectangulaire.

Les fenêtres sont organisées sous forme hiérarchique :

- Chaque application possède au moins une fenêtre, appelée fenêtre principale (root window ou top-level window).
- La fenêtre principale peut être partiellement ou complètement recouverte par des fenêtres dites enfants (child window).
- Les fenêtres enfants peuvent elles aussi être recouvertes par leurs propres fenêtres enfants.

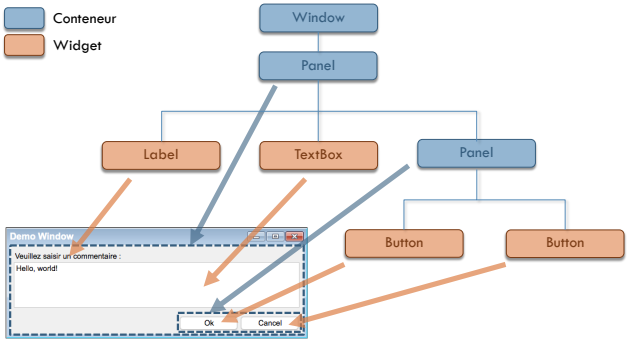
Dans un système d'exploitation moderne, il est virtuellement impossible pour un programme de dessiner en dehors de sa fenêtre principale par accident.

Éléments graphiques de base (*widget* ou *control*)

Dans une hiérarchie de fenêtres, les feuilles représentent les éléments graphiques de base de l'interface utilisateur·rice (bouton, case à cocher, liste déroulante, zone de texte, etc.)

Ces éléments de base sont communément appelés widgets ou contrôles (*control*).

Une fenêtre qui a une ou plusieurs fenêtres enfants est appelée conteneur (*container*).



Bibliothèques de composants

Pour réaliser un programme avec une interface utilisateur·rice graphique, on utilise généralement des bibliothèques spécialisées appelées bibliothèque de composants ou widget toolkits.

Il s'agit le plus souvent de bibliothèque de classes pour des langages orientés objet.

Le terme composant est utilisé pour désigner des classes qui représentent des widgets (liste déroulante, bouton, etc.) ou des conteneurs de widgets.

On peut distinguer deux types de toolkits :

- Toolkit natif : toolkit intégré à une plateforme qui fournit des widgets dits natifs (win32 ou WPF pour Windows, Carbon ou Cocoa pour OSX, Motif pour X11, etc.)
- Toolkit portable : toolkit qui ne dépend pas d'une plateforme particulière (Qt, Swing, HTML, etc.) .

Look and feel

Le look and feel (LAF) est un ensemble de règles qui définit l'aspect visuel (look) et le comportement (feel) des widgets.

L'aspect visuel des widgets dépend du toolkit utilisé (Aero avec win32, Aqua avec Cocoa, etc.) et beaucoup de toolkits permettent de choisir le look (Swing, GTK+, etc.)

Le comportement des widgets dépend également du toolkit utilisé, mais respecte en général les standards de fait établis par les systèmes dominants. Un bouton, une case à cocher ou un bouton radio se comporte sensiblement de la même manière, quel que soit le système.

Vue d'ensemble