CHAPTER 19: MULTIPLE DOCUMENT INTERFACE

MDI (Multiple-Document Interface):

It's a design approach for applications that manage multiple documents in Microsoft Windows.

It allows users to work with several documents simultaneously within a single application window.

Think of it like having multiple tabs open in your web browser, but for different documents within the same program.

Key concepts:

* Parent window: The main application window that holds all the document windows.
* Child windows: The individual document windows that reside within the parent window.
* Client area: The part of the parent window where the child windows are displayed.

How MDI works:

Opening documents: Each document opens in its own child window within the parent window.

Arranging windows: Users can arrange child windows in various ways:

* Tile them horizontally or vertically to view multiple documents side-by-side.
* Cascade them to overlap like a stack of cards.
* Arrange them manually by dragging and resizing.

Switching between windows: Users can switch between documents by clicking on the desired child window, or using keyboard shortcuts.

Managing windows: The application provides features to manage child windows, such as:

* Arranging them (as mentioned above).
* Minimizing, maximizing, or closing them.

Benefits of MDI:

Efficient multitasking: Enables working on multiple documents simultaneously without cluttering the desktop with separate application windows.

Easy comparison: Allows side-by-side comparison of documents for easier referencing and editing.

Organization: Helps keep related documents grouped together within a single application.

Examples of MDI applications:

* Older versions of Microsoft Office applications (Word, Excel, PowerPoint)
* Adobe Photoshop
* Many text editors and code editors

Modern trends:

While MDI was common in older Windows applications, newer applications often use alternative approaches like:

* Tabbed interfaces (similar to web browsers)
* Single-document interfaces (SDI) where each document opens in its own separate window

These approaches offer different user experiences and trade-offs in terms of window management and organization.

MDI in Historical Context:

Early Complexity: While MDI was introduced in Windows 2.0, its implementation was challenging for developers due to intricate programming requirements.

Greater Support: Windows 3.0 and subsequent versions significantly streamlined MDI development by providing built-in support and enhancements.

Key Elements of MDI Applications:

Main Application Window:

Similar to standard application windows, featuring a title bar, menu, sizing border, system menu icon, minimize/maximize/close buttons.

Unique client area, often called the "workspace," designed specifically to house child windows instead of direct program output.

Child Windows (Document Windows):

Resemble small application windows within the main window's workspace.

Possess title bars, sizing borders, system menu icons, minimize/maximize/close buttons, and potentially scroll bars.

Notably lack their own menus, relying on the main application window's menu for actions.

Active Document:

Only one child window can be active at a time, indicated by a highlighted title bar.

Active window appears in front of other child windows, ensuring focus.

Document Containment:

Child windows are confined to the workspace area within the main application window, preventing them from extending beyond its boundaries.

Essential Programming Considerations:

Window Management: MDI support involves specific functions and data structures for managing the relationship between the main application window and its child windows.

Message Handling: MDI applications must process window messages differently to coordinate interactions between the main window and child windows.

User Experience: Programmers must carefully design the MDI interface to ensure intuitive navigation, window management, and document interaction for users.