KEYBOARD ACCELERATORS

Keyboard accelerators are key combinations that allow users to quickly access frequently used commands or actions in a software application.

These shortcuts are typically represented as a combination of two or more keys, often including a modifier key like Ctrl, Alt, or Shift, and a non-modifier key like A, B, C, or D.

Purpose of Keyboard Accelerators

Keyboard accelerators offer several advantages over traditional menu-based navigation:

Increased Efficiency: Keyboard accelerators allow users to perform actions without switching between the keyboard and mouse, significantly improving efficiency and workflow speed.



Reduced Eye Strain: By keeping users focused on the keyboard, keyboard accelerators minimize the need for constant eye movement between the keyboard and the screen, reducing eye strain.



Accessibility Enhancements: Keyboard accelerators provide an alternative input method for users with limited hand or mouse control, enhancing accessibility and usability.



Common Keyboard Accelerator Usage

Keyboard accelerators are widely used in various software applications, including:

* Word Processors: Copy (Ctrl+C), Paste (Ctrl+V), Undo (Ctrl+Z), Redo (Ctrl+Y)
* Web Browsers: Open New Tab (Ctrl+T), Close Tab (Ctrl+W), Switch Tabs (Ctrl+Tab/Ctrl+Shift+Tab), Save Page (Ctrl+S)
* Operating Systems: Cut (Ctrl+X), Copy (Ctrl+C), Paste (Ctrl+V), Undo (Ctrl+Z), Redo (Ctrl+Y), Save (Ctrl+S), Print (Ctrl+P)

Implementing Keyboard Accelerators

Software developers can implement keyboard accelerators using various methods, including:

* Windows API: The Windows API provides functions like TranslateAccelerator and CreateAcceleratorTable to manage keyboard accelerators in Windows applications.
* Cross-Platform Toolkits: Cross-platform toolkits like Qt and GTK+ offer native support for keyboard accelerators, allowing consistent implementation across different platforms.
* Application Frameworks: Application frameworks like .NET Framework and Electron provide built-in functionality for defining and handling keyboard accelerators.

Benefits of Keyboard Accelerators

Keyboard accelerators offer numerous benefits to both users and developers:

* User Benefits: Increased efficiency, reduced eye strain, improved accessibility, enhanced productivity
* Developer Benefits: Simplified code, reduced menu clutter, improved user experience

Encouraging Keyboard Accelerator Use

To encourage users to adopt keyboard accelerators, developers can implement strategies like:

* Prominent Display: Display keyboard shortcuts alongside menu items or provide a dedicated cheat sheet.
* Training and Documentation: Include clear instructions and tutorials on using keyboard accelerators in the application's documentation or help system.
* Customizability: Allow users to customize keyboard shortcuts to suit their preferences and accessibility needs.

Guidelines for Assigning Keyboard Accelerators

Keyboard accelerators, also known as hotkeys, are key combinations that allow users to quickly access frequently used commands or actions in a software application.

When assigning keyboard accelerators, it's crucial to consider consistency, accessibility, and potential conflicts with system functions. Here are some general guidelines to follow:

Consistency with Common Applications: Strive for consistency with keyboard accelerators used in popular applications. This helps users maintain familiarity and avoid confusion when switching between programs.

Avoid Conflicts with Windows Functions: Refrain from using keys like Tab, Enter, Esc, and Spacebar for keyboard accelerators, as these are often reserved for system functions.

Use Modifier Keys Effectively: Utilize modifier keys like Ctrl, Shift, and Alt to create unique and memorable keyboard shortcuts without overloading individual keys.

Consider Old and New Accelerators: When applicable, support both the old and new keyboard accelerators for a specific function, as users may be accustomed to either convention.

Reserve F1 for Help: Dedicate the F1 key to invoke help or context-sensitive assistance.

Avoid F4, F5, and F6: Refrain from using the F4, F5, and F6 keys for keyboard accelerators, as these are often reserved for special functions in Multiple Document Interface (MDI) applications.

Examples of Recommended Keyboard Accelerators

Here's a table of common keyboard accelerators and their associated functions:



THE ACCELERATOR TABLE

An accelerator table is a data structure that defines keyboard shortcuts for menu items and other actions in a Windows application. Each entry in the table specifies an ID, a keystroke combination, and the corresponding menu item or action.

Defining Accelerators in Developer Studio

* You can define accelerator tables using the Accel Properties dialog box in Developer Studio. To create an accelerator table:
* Select the menu item or action for which you want to define a shortcut.
* Right-click and select "Properties" from the context menu.
* In the Properties dialog box, click the "Accel" button.
* In the Accel Properties dialog box, enter a keystroke combination in the "Keystroke" field. You can use virtual key codes, ASCII characters, or a combination of both in conjunction with the Shift, Ctrl, or Alt keys.
* Click "OK" to save the accelerator.
* Loading the Accelerator Table in Your Program

To load an accelerator table into your program, you use the LoadAccelerators function. This function takes two parameters:

* hInstance: The handle to the program's instance.
* lpAcceleratorName: The name of the accelerator table resource. The resource name can be a string or a number.

Here's an example of how to load an accelerator table named MyAccelerators:



Once the accelerator table is loaded, you can use it to translate keystrokes into menu IDs or actions. The TranslateAccelerator function takes three parameters:

* hWindow: The handle to the window that receives the keystroke.
* hMsg: The handle to the message that contains the keystroke.
* wParam: The wParam value of the message.

The TranslateAccelerator function returns a menu ID if the keystroke matches an accelerator in the table. If the keystroke does not match an accelerator, it returns 0.

Here's an example of how to use the TranslateAccelerator function:



If menuID is not 0, it is the ID of the menu item that corresponds to the keystroke. You can then use this ID to perform the corresponding action.

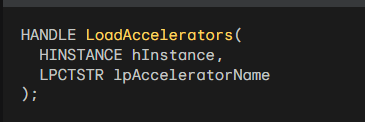
Tips for Defining Accelerators

When defining accelerators, keep the following tips in mind:

* Use consistent keystrokes for similar actions. For example, you might use Ctrl+Z for undo and Ctrl+X for cut.
* Avoid using keystrokes that are already used by Windows. For example, you should not use Ctrl+C for copy, as this is already used by Windows.
* Use descriptive keystrokes. For example, you might use Ctrl+F for find and Ctrl+H for replace.

Loading the Accelerator Table

The LoadAccelerators function is used to load an accelerator table into memory and obtain a handle to it. The syntax of the LoadAccelerators function is as follows:



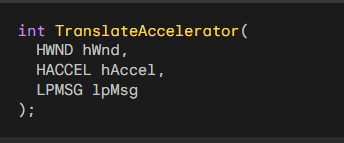
The hInstance parameter is the handle to the program's instance. The lpAcceleratorName parameter is the name of the accelerator table resource. The resource name can be a string or a number.

Here's an example of how to load an accelerator table named MyAccelerators:



Translating Keystrokes

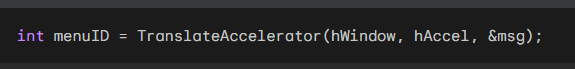
The TranslateAccelerator function is used to translate a keystroke message into a menu ID or action. The syntax of the TranslateAccelerator function is as follows:



The hWnd parameter is the handle to the window that receives the keystroke. The hAccel parameter is the handle to the accelerator table. The lpMsg parameter is a pointer to the message structure that contains the keystroke.

The TranslateAccelerator function returns a menu ID if the keystroke matches an accelerator in the table. If the keystroke does not match an accelerator, it returns 0.

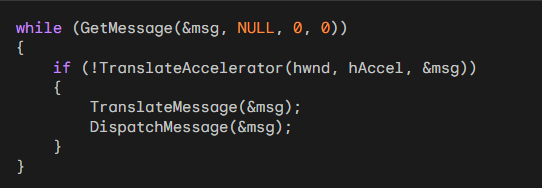
Here's an example of how to use the TranslateAccelerator function:



If menuID is not 0, it is the ID of the menu item that corresponds to the keystroke. You can then use this ID to perform the corresponding action.

Integrating Keyboard Accelerators into the Message Loop

To integrate keyboard accelerators into the message loop, you can modify the standard message loop as follows:



This code will first check whether the keystroke can be translated using the accelerator table. If it can, the TranslateAccelerator function will send the corresponding message to the window procedure. Otherwise, the code will continue with the normal message loop processing.

Understanding the hwnd Parameter

The hwnd parameter is used to specify the window that should receive the keyboard accelerator messages. If you omit the hwnd parameter, the messages will be sent to the window that currently has the input focus.

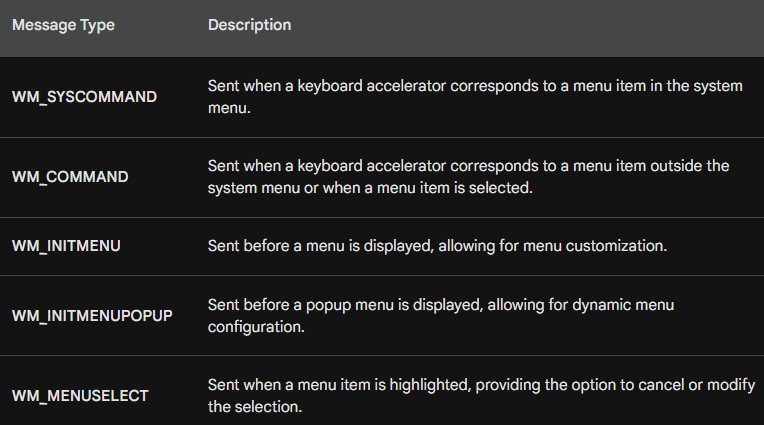
Handling Modal Dialog Boxes and Message Boxes

The TranslateAccelerator function does not translate keyboard messages when a modal dialog box or message box has the input focus. This is because messages for these windows do not come through the program's message loop.

If you want keyboard accelerators to be translated for modal dialog boxes or message boxes, you will need to use a different technique.

One technique is to use the SetWindowsHookEx function to install a hook that intercepts keyboard messages before they are sent to the dialog box or message box.

Types of Accelerator Messages



The TranslateAccelerator function translates keystrokes into corresponding messages, either WM\_SYSCOMMAND or WM\_COMMAND, depending on whether the accelerator corresponds to a menu item in the system menu or not.

WM\_SYSCOMMAND Messages for System Menu Items

When a keyboard accelerator corresponds to a menu item in the system menu, the TranslateAccelerator function sends the window procedure a WM\_SYSCOMMAND message. This message indicates that a system menu command has been invoked using the keyboard.

WM\_COMMAND Messages for Non-System Menu Items

For keyboard accelerators that correspond to menu items outside the system menu, the TranslateAccelerator function sends the window procedure a WM\_COMMAND message. This message indicates that a non-system menu command has been invoked using the keyboard.

WM\_COMMAND Message Parameters

The WM\_COMMAND message contains information about the invoked command, including:

* LOWORD(wParam): The accelerator ID or menu ID of the command.
* HIWORD(wParam): A notification code specific to the command.
* lParam: The handle of the child window control associated with the command, if applicable.

Additional Messages for Menu Items

When a keyboard accelerator corresponds to a menu item, the window procedure also receives the following messages, just as if the menu option had been chosen:

* WM\_INITMENU: Sent before the menu is displayed, allowing for menu customization.
* WM\_INITMENUPOPUP: Sent before a popup menu is displayed, allowing for dynamic menu configuration.
* WM\_MENUSELECT: Sent when a menu item is highlighted, providing the option to cancel or modify the selection.

Handling Disabled Menu Items

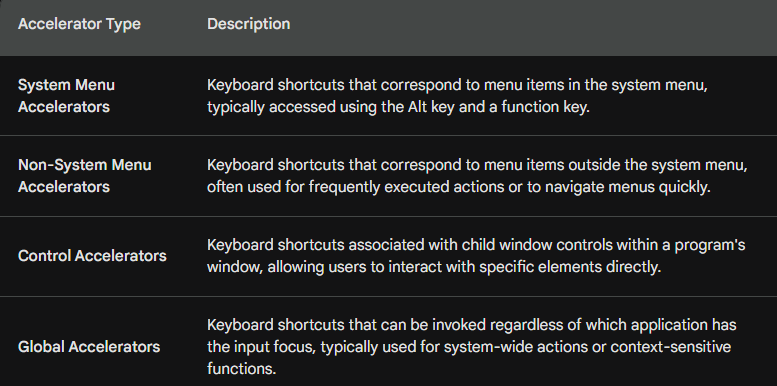
If the keyboard accelerator corresponds to a disabled or grayed menu item, the TranslateAccelerator function does not send the window procedure a WM\_COMMAND or WM\_SYSCOMMAND message. This prevents users from activating unavailable menu options using keyboard shortcuts.

Accelerator Behavior for Minimized Windows

When the active window is minimized, the TranslateAccelerator function sends the window procedure WM\_SYSCOMMAND messages for keyboard accelerators that correspond to enabled system menu items. This allows users to access essential system commands even when the window is minimized.

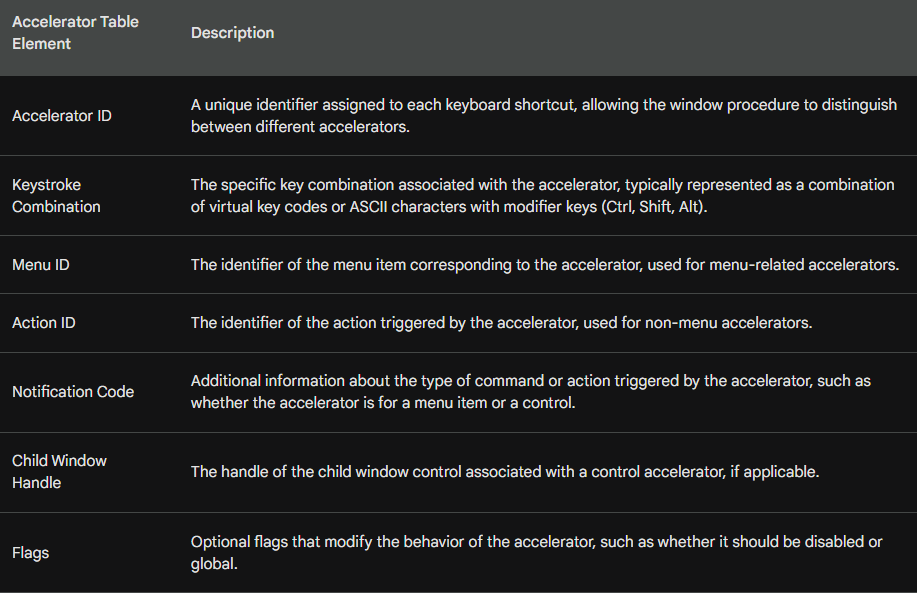
Handling Non-System Menu Accelerators for Minimized Windows

For keyboard accelerators that do not correspond to any menu items, the TranslateAccelerator function sends the window procedure WM\_COMMAND messages even when the window is minimized. This ensures that users can still access other commands using keyboard shortcuts.



Additional Points:

* Accelerator IDs are unique identifiers assigned to each keyboard shortcut, allowing the window procedure to distinguish between different accelerators.
* Notification codes provide additional information about the type of command or action triggered by the accelerator.
* Child window handles identify the specific control associated with a control accelerator.
* Global accelerators are registered using the RegisterHotKey function and require elevated privileges in some cases.



In summary, the TranslateAccelerator function plays a crucial role in translating keystrokes into corresponding messages, enabling users to efficiently interact with applications using keyboard shortcuts.

*Popad2 program in chapter 10 folder….*

POPPAD2: A Rudimentary Notepad with Menus and Accelerators

This document details the POPPAD2 program, a rudimentary notepad application that builds upon the previous POPPAD1 program. POPPAD2 introduces several new functionalities, including:

Menus: The program incorporates File and Edit menus, providing access to various options like New, Open, Save, Undo, Cut, Copy, Paste, Clear, and Select All.

Accelerators: Keyboard shortcuts are associated with specific menu items, enabling faster access to frequently used functions.

Edit Control Functionality: POPPAD2 utilizes a child window edit control to handle text editing, offering features like undo, cut, copy, paste, clear, and select all.

Functionality Breakdown

Menus:

*File Menu:*

Currently, these options are non-functional and will generate a beep sound upon selection.

Future chapters will implement functionalities for New, Open, Save, Save As, and Print.

*Edit Menu:*

Undo: Sends a WM\_UNDO message to the child window edit control, enabling undo functionality.

Cut: Sends a WM\_CUT message to the edit control, copying the selected text to the clipboard and removing it from the document.

Copy: Sends a WM\_COPY message to the edit control, copying the selected text to the clipboard without removing it from the document.

Paste: Sends a WM\_PASTE message to the edit control, inserting the text from the clipboard into the document at the current cursor position.

Clear: Sends a WM\_CLEAR message to the edit control, deleting all text from the document.

Select All: Sends a EM\_SETSEL message to the edit control, selecting all text in the document.

*Additional Features:*

The program dynamically updates the enabled state of menu items based on the current context. For example, the "Cut" and "Copy" options are only enabled when there is text selected.

Keyboard shortcuts provide quick access to specific menu items, improving user experience and efficiency.

The program confirms with the user before closing, ensuring any unsaved data is not lost accidentally.

POPPAD2.RC:

This file defines the program's resources, including icons, menus, and keyboard shortcuts.

*Icons:*

The file specifies the use of "poppad2.ico" as the program's icon.

*Menus:*

The POPPAD2 menu consists of three main categories: File, Edit, and Help.

*File Menu:*

This menu provides options for creating new files (IDM\_FILE\_NEW), opening existing files (IDM\_FILE\_OPEN), saving files (IDM\_FILE\_SAVE), saving files with a different name (IDM\_FILE\_SAVE\_AS), printing (IDM\_FILE\_PRINT), and exiting the program (IDM\_APP\_EXIT).

*Edit Menu:*

This menu offers options for undoing actions (IDM\_EDIT\_UNDO), cut (IDM\_EDIT\_CUT), copy (IDM\_EDIT\_COPY), paste (IDM\_EDIT\_PASTE), deleting text (IDM\_EDIT\_CLEAR), and selecting all text (IDM\_EDIT\_SELECT\_ALL).

*Help Menu:*

This menu provides access to help information (IDM\_HELP\_HELP) and an about dialogue for the program (IDM\_APP\_ABOUT).

*Accelerators:*

The POPPAD2 resource file defines keyboard shortcuts for various functions:

Undo: Ctrl+Z (IDM\_EDIT\_UNDO)

Delete: Del (IDM\_EDIT\_CLEAR)

Cut: Ctrl+X, Shift+Del (IDM\_EDIT\_CUT)

Help: F1 (IDM\_HELP\_HELP)

Copy: Ctrl+Insert (IDM\_EDIT\_COPY)

Paste: Shift+Insert, Ctrl+V (IDM\_EDIT\_PASTE)

Select All: Ctrl+A (IDM\_EDIT\_SELECT\_ALL)

RESOURCE.H:

This file provides symbolic names for the various menu options and keyboard shortcuts defined in POPPAD2.RC.

This allows the application code to refer to these resources by name instead of using numerical identifiers, making the code more readable and maintainable.

