MENUS AND OTHER RESOURCES

Icons

Icons are small graphical images that represent programs, files, or folders. They are displayed in the title bar of application windows, in the Start menu, in the taskbar, in Windows Explorer, and as shortcuts on the desktop. Icons can be in color or black and white, and they can be in any size.

Cursors

Cursors are graphical images that represent the mouse pointer. They change shape depending on the context, such as when the mouse is hovering over a link, when it is selecting text, or when it is resizing a window. Cursors can be in color or black and white, and they can be in any size.

Character Strings

Character strings are text strings that are used by programs. They can be used for menus, dialog boxes, error messages, and other purposes. Character strings can be stored in the program's .EXE file or in a separate resource file.

Custom Resources

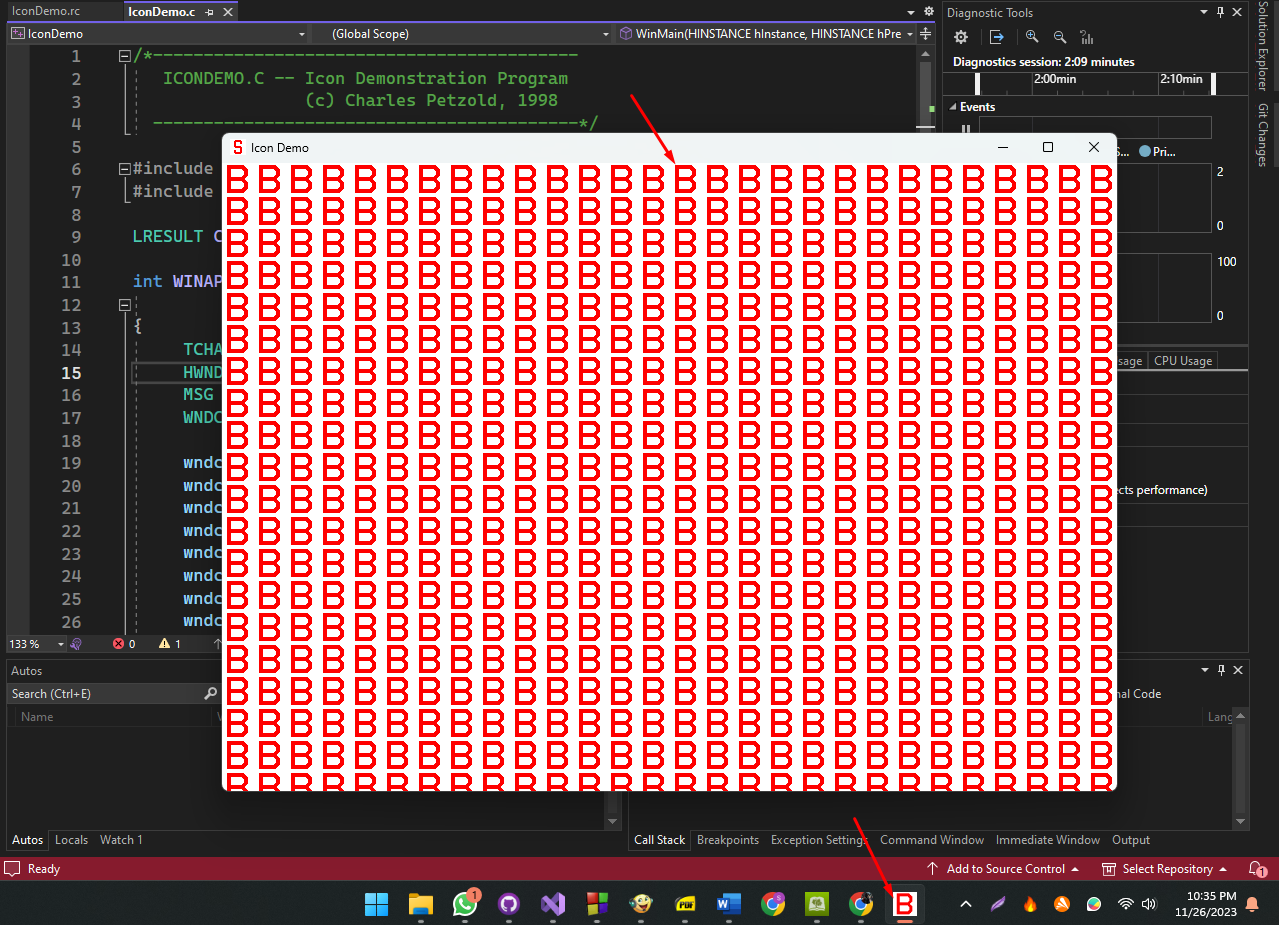
Custom resources are any type of resource that is not an icon, cursor, character string, or menu. They can be used for storing data that is specific to the program, such as images, sounds, or video. Custom resources are stored in the program's .EXE file or in a separate resource file.

Menus

Menus are hierarchical lists of options that users can select to perform actions in a program. They can be displayed as pull-down menus, context menus, or toolbars. Menus can be customized to include the specific options that a program needs.

Keyboard Accelerators

Keyboard accelerators are keyboard shortcuts that allow users to quickly perform actions in a program. They are typically combinations of two or more keys, such as Ctrl+S to save a file. Keyboard accelerators can be customized to the user's preferences.



Introduction

Resources in C++ programming offer a convenient way to bind various components of a program into the executable file.

This eliminates the need for separate files, making it easier to manage and distribute the application. For instance, icons, cursors, strings, and other custom resources can be included within the program's .EXE file.

Icons as Resources

One notable example is the inclusion of icons. Typically, an icon would require a separate file, but with resources, it can be stored in an editable file on the developer's computer and bound into the .EXE during the build process.

This approach streamlines development and ensures that the icon is an integral part of the executable.

Adding an Icon to a Program

To add an icon to a program, Visual C++ Developer Studio provides the Image Editor, allowing developers to draw an icon that gets saved in an .ICO file.

Simultaneously, Developer Studio generates a resource script (with .RC extension) listing all program resources and a header file (RESOURCE.H) enabling the program to reference these resources.

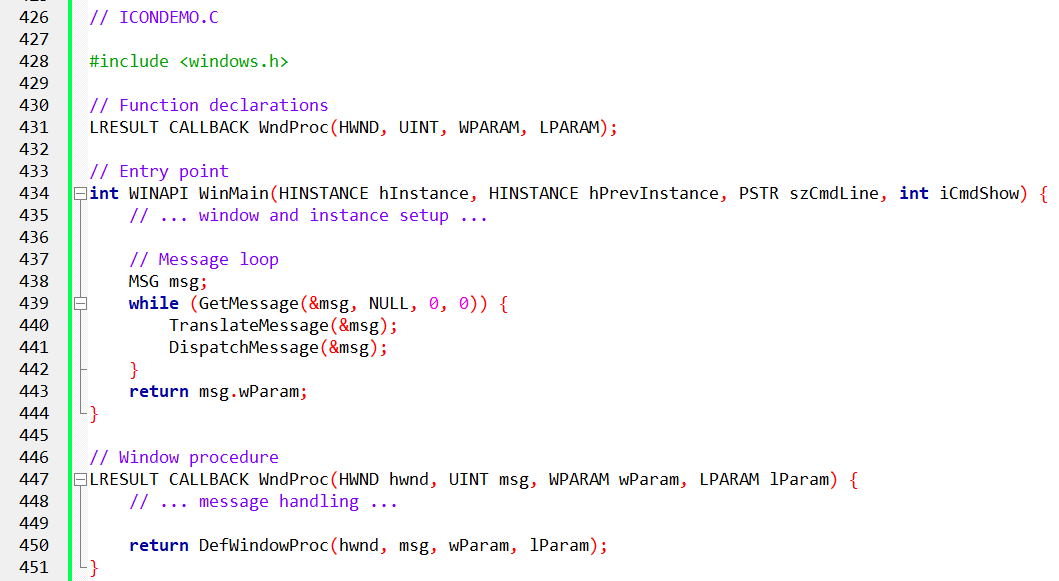
Project Setup: ICONDEMO

Let's illustrate this process by creating a new project named ICONDEMO in Visual C++ Developer Studio.

After creating the project, the studio generates several files, including ICONDEMO.DSW, ICONDEMO.DSP, and ICONDEMO.MAK. Additionally, a C source code file (ICONDEMO.C) is created, where the program logic will be implemented.

Example Program Structure

Here's a simplified version of the program structure:



The source code is in the icondemo folder….

ICONDEMO.C is a Windows program that demonstrates the use of icons in a graphical user interface (GUI) application. It creates a window and fills it with copies of an icon specified in the program's resources.

Windows Header File: The #include <windows.h> statement includes the Windows header file, which contains essential definitions for interacting with the Windows API.

Resource File Inclusion: The #include "resource.h" statement incorporates the resource file, which holds the program's resources, including icons and cursors.

Window Procedure Function: The LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam) function serves as the window procedure, responsible for handling messages sent to the window by the operating system.

Program Entry Point: The int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR szCmdLine, int iCmdShow) function acts as the program's entry point, executed when the program starts.

Variable Declarations: Variables are declared to store essential program information, including the program's name (szAppName), window handle (hwnd), current message (msg), and window class structure (wndclass).

Window Class Configuration: The window class structure (wndclass) is configured with settings that define the window's appearance and behavior.

Window Registration: The RegisterClass (&wndclass) statement registers the window class with the system, allowing the program to create windows based on that class.

Window Creation: The hwnd = CreateWindow (...) statement creates a window using the registered window class, specifying the window's name, position, size, and other attributes.

Window Display: The ShowWindow (hwnd, iCmdShow) function displays the created window, making it visible to the user.

Window Update: The UpdateWindow (hwnd) function refreshes the window's contents, ensuring it is properly rendered on the screen.

Message Loop: The while (GetMessage (&msg, NULL, 0, 0)) (...) loop continuously retrieves messages from the message queue and dispatches them to the window procedure function.

Message Translation: The TranslateMessage (&msg) statement translates the retrieved message into a format compatible with the window procedure function.

Message Dispatching: The DispatchMessage (&msg) statement sends the translated message to the window procedure function for processing.