Eigenschaften der GUI-Elemente

CButton GUI-Overklasse

- EnableWindow(bool); // enabled des GUI-Elements
-
- ShowWindow(SW_SHOW) // sichtbar
- ShowWindow(SW_HIDE) // unsichtbar
-
- SetCheck(bool) // setzte / löscht den Status eines radiobuttons

CStatic

- Align Text
- No Wrap
- Simple
- Static Edge
- Sunken
- Visible // ShowWindow

Hinweis:

Alle Label-Elemente haben die defaultmäßig die ID: IDC_STATIC
Damit kann man keine Membervariable einrichten (doppelte Namen)
Erst umbenennen

CEdit

- Align Text
- Border
- Lowercase
- Uppercase
- Number
- Multiline
- Password
- Read Only
- Visible // ShowWindow

CRadiobutton

- Bitmap
- Flat
- Multi Line
- Static Edge
- Visible // ShowWindow

CCheckbox

- Bitmap
- Flat
- Multi Line
- Right Align Text
- Right to Left Reading Order
- Static Edge
- Visible // ShowWindow

CSpinButtonCtrl

- Arrow Keys
- No Thousands
- Orientation
- Static Edge
- Wrap
- Auto Buddy
- Set Buddy Integer
- Visible // ShowWindow

- Einfügen eines Drehfeldes
- Einfügen eines Editorfeldes
- Ändern der Number-Eigenschaften im Editorfeld
- Sofort danach Einfügen des Drehfeldes
- Ändern der beiden Buddy-Eigenschaften im Register „Formate“
- Ändern der ID auf z. B. IDC_SPIN_XTICKS
- Damit ist die Verknüpfung zum Editorfeld definiert
- Leider ist die Reihenfolge (Decrement, Increment) falsch

Abhilfe:

Dazu muss in der OnInitDialog-Methode folgender Code eingefügt werden:

```
m_SpinEdit.SetRange(0,12);     // (Membervariable Control)
```

CComboBox

- Lowercase
- Uppercase
- Vertical Scrollbar
- Type Dropdown (list+Eingabe)
  Dropdownlist (nur liste)
- Data Semikolon als Trenner
- Sort
- Visible // ShowWindow
- GetCurSel() // Membervariable Control
- SetCurSel(int index) // Membervariable Control

Init einer Listbox:

In InitDialog:
    Hinter // TODO: Add extra initialization here
        m_combo_ctrl.AddString(_T("44567"));
        m_combo_ctrl.SetCurSel(2);

Slider

Methoden:

```
Int GetPos()
void SetPos(int i);
SetPageSize(int i);        // Anzahl der Einheiten pro Mausklick
GetTic                     // Aufbau der Skala
GetTicPos                  // Aufbau der Skala
GetTicArray                // Aufbau der Skala
GetNumTicks                // Aufbau der Skala
ClearSel                   // Selektion löschen
```

Einbau einer Slider-Change-Event

In der Headerdatei:
als public
afx_msg void OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);

In der Dlg.cpp

```
void CPagerCtrlPage::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    int nBorderSize = m_sliderBorderSize.GetPos();
    CString sStr;
    sStr.Format(_T("%d"), nBorderSize);
    m_staticBorderSize.SetWindowText(sStr);
    AfxMessageBox(sStr);
}
```

In der Dlg.cpp

```
BEGIN_MESSAGE_MAP(CTestSliderDlg, CDialog)
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    //}}AFX_MSG_MAP
    ON_WM_HSCROLL()
END_MESSAGE_MAP()
```

nSBCode:	Aktionscode:
TB_TOP	Pos1 gedrückt
TB_BOTTOM	Ende gedrückt
TB_LINEDOWN	Taste Pfeil links oder rechts
TB_PAGEDOWN	Taste Page Down gedrückt
TB_PAGEUP	Taste Page Up gedrückt
TB_THUMBTRACK	Regler wird mit der Maus gezogen
TB_THUMBPOSITION	Linke Maustaste wurde nach dem Ziehen losgelassen
TB_ENDTRACK	Taste oder Maustaste wurde nach dem Ziehen losgelassen

Beispiele

http://msdn.microsoft.com/en-us/library/ms386053(VS.71).aspx

Klassen und GUI-Elemente

Control	MFC class	Description
animation	CAnimateCtrl	Displays successive frames of an AVI video clip
button	CButton	Pushbuttons that cause an action; also used for check boxes, radio buttons, and group boxes
combo box	CComboBox	Combination of an edit box and a list box
date and time picker	CDateTimeCtrl	Allows the user to choose a specific date or time value
edit box	CEdit	Boxes for entering text
extended combo box	CComboBoxEx	A combo box control with the ability to display images
header	CHeaderCtrl	Button that appears above a column of text; controls width of text displayed
hotkey	CHotKeyCtrl	Window that enables user to create a "hot key" to perform an action quickly
image list	CImageList	Collection of images used to manage large sets of icons or bitmaps (image list isn't really a control; it supports lists used by other controls)
list (Explorer)	CListCtrl	Window that displays a list of text with icons
list box	CListBox	Box that contains a list of strings
month calendar	CMonthCalCtrl	Control that displays date information
progress	CProgressCtrl	Window that indicates progress of a long operation
rebar	CRebarCtrl	Tool bar that can contain additional child windows in the form of controls
rich edit	CRichEditCtrl	Window in which user can edit with character and paragraph formatting (see Classes Related to Rich Edit Controls)
scroll bar	CScrollBar	Scroll bar used as a control inside a dialog box (not on a window)
slider	CSliderCtrl	Window containing a slider control with optional tick marks
spin button	CSpinButtonCtrl	Pair of arrow buttons user can click to increment or decrement a value
static-text	CStatic	Text for labeling other controls
status bar	CStatusBarCtrl	Window for displaying status information, similar to MFC class CStatusBar
tab	CTabCtrl	Analogous to the dividers in a notebook; used in "tab dialog boxes" or property sheets
toolbar	CToolBarCtrl	Window with command-generating buttons, similar to MFC class CToolBar
tool tip	CToolTipCtrl	Small pop-up window that describes purpose of a toolbar button or other tool
tree	CTreeCtrl	Window that displays a hierarchical list of items

CList Box

- Border
- Group
- MultiColumn
- Selection
- Sort
- Vertical Scrollbar
- Visible // ShowWindow
- GetCurSel() // Membervariable Control
- SetCurSel(int index) // Membervariable Control

Init einer Listbox:

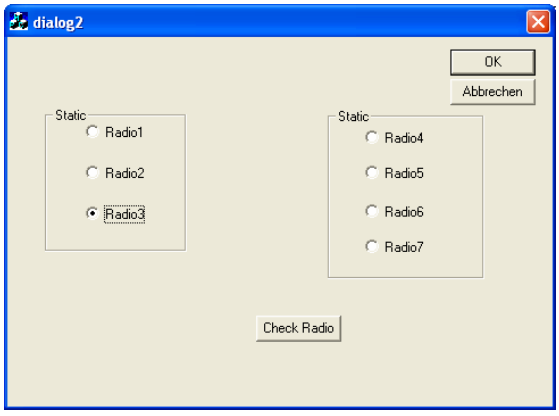
In InitDialog:

Hinter // TODO: Add extra initialization here
m_liste_ctrl.AddString(_T("IB"));

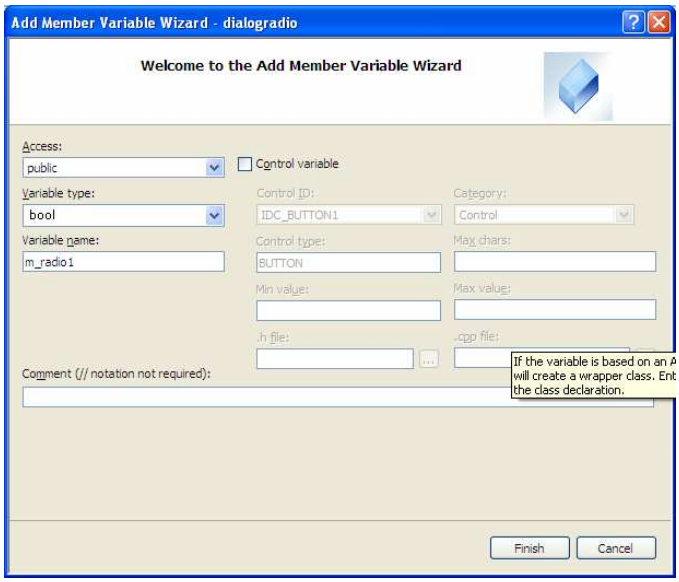
UpdateData(bRichtung)

- TRUE ⇒ Aus GUI nach Membervariable
- FALSE ⇒ Aus Membervariable nach GUI

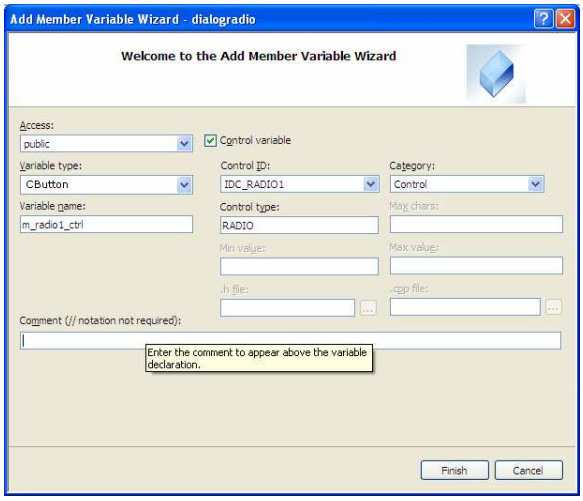
2. Übung



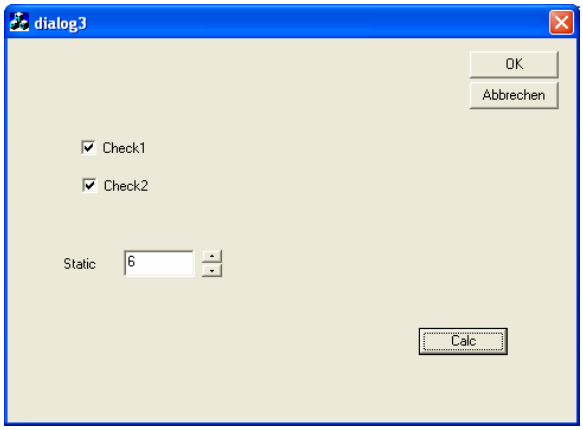
Einfügen einer Member-Variablen Wert



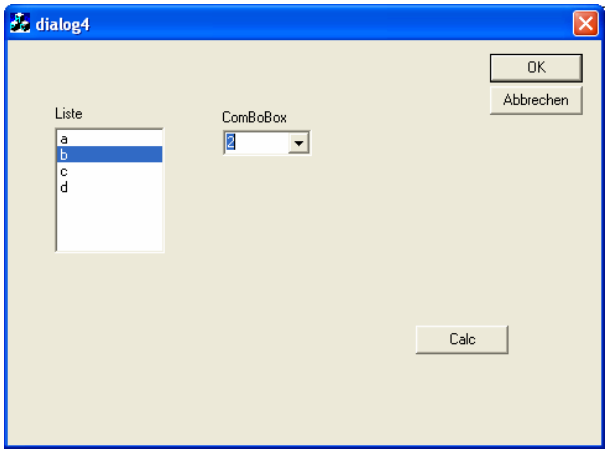
Einfügen einer Member-Variablen Control



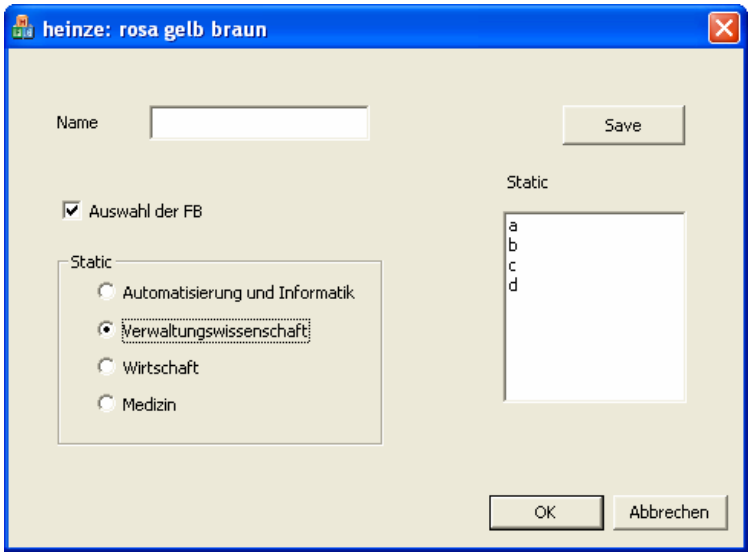
3. Übung



4. Beispiel



5. Übung



Checkbox kontrolliert enabled der Radiobuttons

```
void CheinzeDlg::OnPaint()
{
   if (IsIconic())
   {
      CPaintDC dc(this); // Gerätekontext zum Zeichnen

      SendMessage(WM_ICONERASEBKGND, reinterpret_cast<WPARAM>
                  (dc.GetSafeHdc()), 0);

      // Symbol in Clientrechteck zentrieren
      int cxIcon = GetSystemMetrics(SM_CXICON);
      int cyIcon = GetSystemMetrics(SM_CYICON);
      CRect rect;
      GetClientRect(&rect);
      int x = (rect.Width() - cxIcon + 1) / 2;
      int y = (rect.Height() - cyIcon + 1) / 2;

      // Symbol zeichnen
      dc.DrawIcon(x, y, m_hIcon);
   }
   else
   {
      CPaintDC dc(this); // Gerätekontext zum Zeichnen
      CRect rect;
      GetClientRect(&rect);
      CBrush brush1( RGB(255,0,255) );
      dc.SelectObject(&brush1);
      dc.Rectangle(&rect);

      CDialog::OnPaint();
   }
}
```

2. Übung

```
void CDialog2Dlg::OnBnCalc()
{
      int i1, i2;
      UpdateData(true);
      i1 = m_Radio1;
      i2 = m_Radio2;
      CString sStr;
      sStr.Format("i1: %d\ni2: %d",i1,i2);
      AfxMessageBox(sStr);
}

      int iButton = GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO3) - IDC_RADIO1;
      CString sStr;
      sStr.Format(_T("Radiowert: %d"),iButton);
      AfxMessageBox( sStr );


void CdialogradioDlg::OnBnClickedCheck1()
{
      // TODO: Add your control notification handler code here
    CButton * pBtn1 = (CButton *) GetDlgItem(IDC_RADIO1);
    CButton * pBtn2 = (CButton *) GetDlgItem(IDC_RADIO2);
    CButton * pBtn3 = (CButton *) GetDlgItem(IDC_RADIO3);
      UpdateData(true);
      // nur eine
    if (m_checkbox) {
        pBtn1->EnableWindow(true);
        pBtn2->EnableWindow(true);
        pBtn3->EnableWindow(true);
       }
       else {
        pBtn1->EnableWindow(false);
        pBtn2->EnableWindow(false);
        pBtn3->EnableWindow(false);
        //m_radio1_ctrl.EnableWindow(false);
       }

}
```

3. Übung

```
void CDialog3Dlg::OnBnCalc()
{
      BOOL b1, b2;
      int zahl;
      UpdateData(true);
```

```
    b1 = m_Check1;
    b2 = m_Check2;
    zahl = m_Zahl;
    CString sStr;
    sStr.Format("Check1: %d\nCheck2: %d\nZahl: %d",b1,b2,zahl);
    AfxMessageBox(sStr);


}
```

4. Übung

InitDialog

```
  CListBox * pListe1 = (CListBox *) GetDlgItem(IDC_LISTE1);
  CComboBox * pListe2 = (CComboBox *) GetDlgItem(IDC_LISTE2);

      pListe1->AddString("a");
      pListe1->AddString("b");
      pListe1->AddString("c");
      pListe1->AddString("d");

      // CListBoxm_liste1_ctrl;
      m_liste1_ctrl.AddString("3333");

      pListe2->AddString("1");
      pListe2->AddString("2");
      pListe2->AddString("3");
      pListe2->AddString("4");
      pListe2->AddString("5");
      pListe2->AddString("6");

      CDialog::OnInitDialog();
```

Membervariablen

- m_liste1
- m_liste2

void CDialog4Dlg::OnBnCalc()

```
{
    UpdateData(true);
    CString sStr1;
    CString sStr2;
    sStr1 = m_Liste1;
    sStr2 = m_Liste2;
    AfxMessageBox(sStr1 + "\n" + sStr2);
}
```

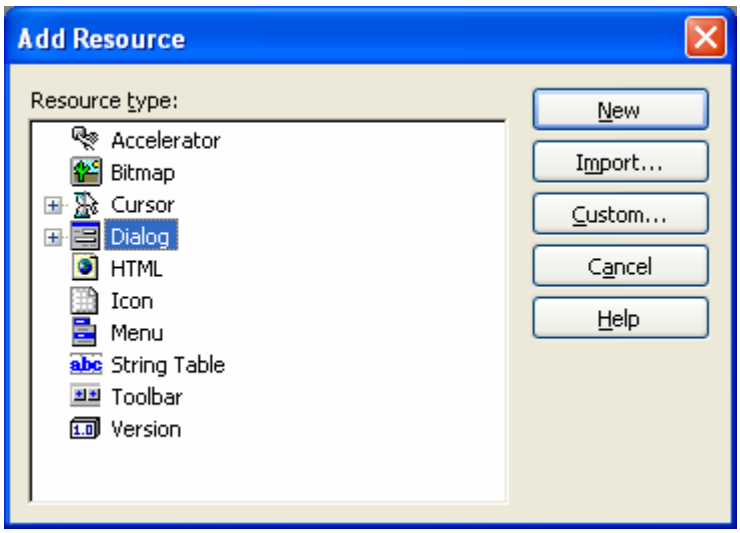
Neuer Modale Dialog

Dialog-Erstellen

Recourcen Register

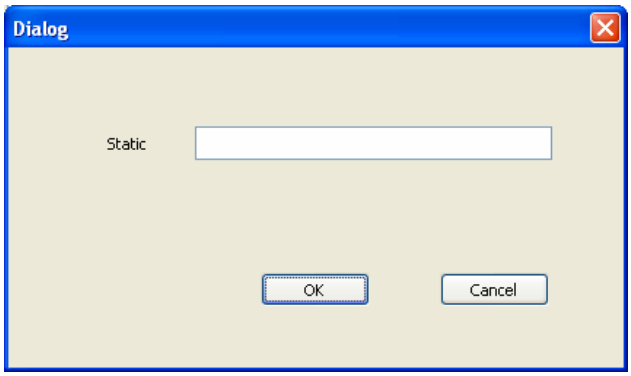
Dialog Eintrag

Add Dialog



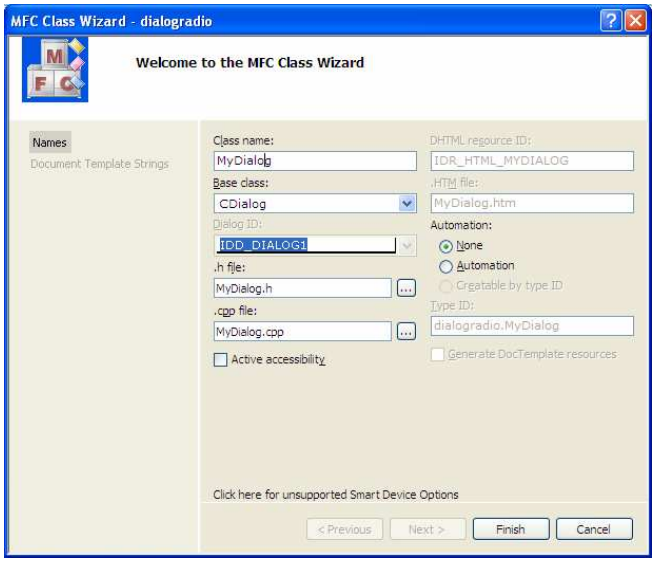
Schalter New

Eintragen der GUI-Elemente



Erzeugen einer Klasse

- Ins Dialogfenster klicken



Wechseln ins erste Dialogfenster

Eintragen eines Schalters myDialog
OnClick Event erzeugen

#include "MyDialog.h"

Eintragen in Click

```
void CdialogradioDlg::OnBnClickedButton3()
{
  MyDialog dlg;
      //dlg.m_Label = "Radius";
      //dlg.m_Zahl = X;
      if (dlg.DoModal() == IDOK) {
            //X = dlg.m_Zahl;
            //pDoc->setX(X);
            //pDoc->UpdateAllViews( this );    // oder NULL
            //Invalidate();
      }  // if
}
```

member-variablen definieren
Vorher setzen, nachher auslesen

```
void CdialogradioDlg::OnBnClickedButton3()
{
  MyDialog dlg;
      dlg.m_edit = _T("Bitte Namen eingeben");
      if (dlg.DoModal() == IDOK) {
            AfxMessageBox(dlg.m_edit);
      }  // if
      else {
            AfxMessageBox(_T("Schade, dass Sie abgebrochen haben") );
      }
}
```

isWindowVisible

CWnd Class Members

Initialization	Dialog-Box Item Functions	Initialization Message Handlers
Window State Functions	Data-Binding Functions	System Message Handlers
Window Size and Position	Menu Functions	General Message Handlers
Window Access Functions	Tool Tip Functions	Control Message Handlers
Update/Painting Functions	Timer Functions	Input Message Handlers
Coordinate Mapping Functions	Alert Functions	Nonclient-Area Message Handlers
Window Text Functions	Window Message Functions	MDI Message Handlers
Scrolling Functions	Clipboard Functions	Clipboard Message Handlers
Drag-Drop Functions	OLE Controls	Menu Loop Notification
Caret Functions	Overridables	

Data Members

m_hWnd	Indicates the **HWND** attached to this **CWnd**.

Construction/Destruction

CWnd	Constructs a **CWnd** object.
DestroyWindow	Destroys the attached Windows window.

Initialization

Create	Creates and initializes the child window associated with the **CWnd** object.
PreCreateWindow	Called before the creation of the Windows window attached to this **CWnd** object.
CalcWindowRect	Called to calculate the window rectangle from the client rectangle.
GetStyle	Returns the current window style.

GetExStyle	Returns the window's extended style.
Attach	Attaches a Windows handle to a **CWnd** object.
Detach	Detaches a Windows handle from a **CWnd** object and returns the handle.
PreSubclassWindow	Allows other necessary subclassing to occur before SubclassWindow is called.
SubclassWindow	Attaches a window to a **CWnd** object and makes it route messages through the **CWnd**'s message map.
UnsubclassWindow	Detaches a window from a **CWnd** object
FromHandle	Returns a pointer to a **CWnd** object when given a handle to a window. If a **CWnd** object is not attached to the handle, a temporary **CWnd** object is created and attached.
FromHandlePermanent	Returns a pointer to a **CWnd** object when given a handle to a window. If a **CWnd** object is not attached to the handle, **NULL** is returned.
DeleteTempMap	Called automatically by the **CWinApp** idle-time handler and deletes any temporary **CWnd** objects created by **FromHandle**.
GetSafeHwnd	Returns **m_hWnd**, or **NULL** if the **this** pointer is **NULL**.
CreateEx	Creates a Windows overlapped, pop-up, or child window and attaches it to a **CWnd** object.
CreateControl	Create an OLE control that will be represented in an MFC program by a **CWnd** object.

Window State Functions

IsWindowEnabled	Determines whether the window is enabled for mouse and keyboard input.
EnableWindow	Enables or disables mouse and keyboard input.
GetActiveWindow	Retrieves the active window.
SetActiveWindow	Activates the window.
GetCapture	Retrieves the **CWnd** that has the mouse capture.
SetCapture	Causes all subsequent mouse input to be sent to the **CWnd**.
GetFocus	Retrieves the **CWnd** that currently has the input focus.
SetFocus	Claims the input focus.

GetDesktopWindow	Retrieves the Windows desktop window.
GetForegroundWindow	Returns a pointer to the foreground window (the top-level window with which the user is currently working).
SetForegroundWindow	Puts the thread that created the window into the foreground and activates the window.
GetIcon	Retrieves the handle to an icon.
SetIcon	Sets the handle to a specific icon.
GetWindowContextHelpId	Retrieves the help context identifier.
SetWindowContextHelpId	Sets the help context identifier.
ModifyStyle	Modifies the current window style.
ModifyStyleEx	Modifies the window's extended style.

Window Size and Position

GetWindowPlacement	Retrieves the show state and the normal (restored), minimized, and maximized positions of a window.
SetWindowPlacement	Sets the show state and the normal (restored), minimized, and maximized positions for a window.
GetWindowRgn	Retrieves a copy of the window region of a window.
SetWindowRgn	Sets the region of a window.
IsIconic	Determines whether **CWnd** is minimized (iconic).
IsZoomed	Determines whether **CWnd** is maximized.
MoveWindow	Changes the position and dimensions of **CWnd**.
SetWindowPos	Changes the size, position, and ordering of child, pop-up, and top-level windows.
ArrangeIconicWindows	Arranges all the minimized (iconic) child windows.
BringWindowToTop	Brings **CWnd** to the top of a stack of overlapping windows.
GetWindowRect	Gets the screen coordinates of **CWnd**.
GetClientRect	Gets the dimensions of the **CWnd** client area.

Window Access Functions

ChildWindowFromPoint	Determines which, if any, of the child windows contains the specified point.

FindWindow	Returns the handle of the window, which is identified by its window name and window class.
GetNextWindow	Returns the next (or previous) window in the window manager's list.
GetOwner	Retrieves a pointer to the owner of a **CWnd**.
SetOwner	Changes the owner of a **CWnd**.
GetTopWindow	Returns the first child window that belongs to the **CWnd**.
GetWindow	Returns the window with the specified relationship to this window.
GetLastActivePopup	Determines which pop-up window owned by **CWnd** was most recently active.
IsChild	Indicates whether **CWnd** is a child window or other direct descendant of the specified window.
GetParent	Retrieves the parent window of **CWnd** (if any).
GetSafeOwner	Retrieves the safe owner for the given window.
SetParent	Changes the parent window.
WindowFromPoint	Identifies the window that contains the given point.
GetDlgItem	Retrieves the control with the specified ID from the specified dialog box.
GetDlgCtrlID	If the **CWnd** is a child window, calling this function returns its ID value.
SetDlgCtrlID	Sets the window or control ID for the window (which can be any child window, not only a control in a dialog box).
GetDescendantWindow	Searches all descendant windows and returns the window with the specified ID.
GetParentFrame	Retrieves the **CWnd** object's parent frame window.
SendMessageToDescendants	Sends a message to all descendant windows of the window.
GetTopLevelParent	Retrieves the window's top-level parent.
GetTopLevelOwner	Retrieves the top-level window.
GetParentOwner	Returns a pointer to a child window's parent window.
GetTopLevelFrame	Retrieves the window's top-level frame window.
UpdateDialogControls	Call to update the state of dialog buttons and other controls.

UpdateData	Initializes or retrieves data from a dialog box.
CenterWindow	Centers a window relative to its parent.

Update/Painting Functions

BeginPaint	Prepares **CWnd** for painting.
EndPaint	Marks the end of painting.
Print	Draws the current window in the specified device context.
PrintClient	Draws any window in the specified device context (usually a printer device context).
LockWindowUpdate	Disables or reenables drawing in the given window.
UnlockWindowUpdate	Unlocks a window that was locked with **CWnd::LockWindowUpdate**.
GetDC	Retrieves a display context for the client area.
GetDCEx	Retrieves a display context for the client area, and enables clipping while drawing.
RedrawWindow	Updates the specified rectangle or region in the client area.
GetWindowDC	Retrieves the display context for the whole window, including the caption bar, menus, and scroll bars.
ReleaseDC	Releases client and window device contexts, freeing them for use by other applications.
UpdateWindow	Updates the client area.
SetRedraw	Allows changes in **CWnd** to be redrawn or prevents changes from being redrawn.
GetUpdateRect	Retrieves the coordinates of the smallest rectangle that completely encloses the **CWnd** update region.
GetUpdateRgn	Retrieves the **CWnd** update region.
Invalidate	Invalidates the entire client area.
InvalidateRect	Invalidates the client area within the given rectangle by adding that rectangle to the current update region.
InvalidateRgn	Invalidates the client area within the given region by adding that region to the current update region.
ValidateRect	Validates the client area within the given rectangle by removing the rectangle from the current update region.

ValidateRgn	Validates the client area within the given region by removing the region from the current update region.
ShowWindow	Shows or hides the window.
IsWindowVisible	Determines whether the window is visible.
ShowOwnedPopups	Shows or hides all pop-up windows owned by the window.
EnableScrollBar	Enables or disables one or both arrows of a scroll bar.

Coordinate Mapping Functions

MapWindowPoints	Converts (maps) a set of points from the coordinate space of the **CWnd** to the coordinate space of another window.
ClientToScreen	Converts the client coordinates of a given point or rectangle on the display to screen coordinates.
ScreenToClient	Converts the screen coordinates of a given point or rectangle on the display to client coordinates.

Window Text Functions

SetWindowText	Sets the window text or caption title (if it has one) to the specified text.
GetWindowText	Returns the window text or caption title (if it has one).
GetWindowTextLength	Returns the length of the window's text or caption title.
SetFont	Sets the current font.
GetFont	Retrieves the current font.

Scrolling Functions

GetScrollPos	Retrieves the current position of a scroll box.
GetScrollRange	Copies the current minimum and maximum scroll-bar positions for the given scroll bar.
ScrollWindow	Scrolls the contents of the client area.
ScrollWindowEx	Scrolls the contents of the client area. Similar to **ScrollWindow**, with additional features.
GetScrollInfo	Retrieves the information that the **SCROLLINFO** structure maintains about a scroll bar.

GetScrollLimit	Retrieves the limit of the scroll bar.
SetScrollInfo	Sets information about the scroll bar.
SetScrollPos	Sets the current position of a scroll box and, if specified, redraws the scroll bar to reflect the new position.
SetScrollRange	Sets minimum and maximum position values for the given scroll bar.
ShowScrollBar	Displays or hides a scroll bar.
EnableScrollBarCtrl	Enables or disables a sibling scroll-bar control.
GetScrollBarCtrl	Returns a sibling scroll-bar control.
RepositionBars	Repositions control bars in the client area.

Drag-Drop Functions

DragAcceptFiles	Indicates the window will accept dragged files.

Caret Functions

CreateCaret	Creates a new shape for the system caret and gets ownership of the caret.
CreateSolidCaret	Creates a solid block for the system caret and gets ownership of the caret.
CreateGrayCaret	Creates a gray block for the system caret and gets ownership of the caret.
GetCaretPos	Retrieves the client coordinates of the caret's current position.
SetCaretPos	Moves the caret to a specified position.
HideCaret	Hides the caret by removing it from the display screen.
ShowCaret	Shows the caret on the display at the caret's current position. Once shown, the caret begins flashing automatically.

Dialog-Box Item Functions

CheckDlgButton	Places a check mark next to or removes a check mark from a button control.
CheckRadioButton	Checks the specified radio button and removes the

	check mark from all other radio buttons in the specified group of buttons.
GetCheckedRadioButton	Returns the ID of the currently checked radio button in a group of buttons.
DlgDirList	Fills a list box with a file or directory listing.
DlgDirListComboBox	Fills the list box of a combo box with a file or directory listing.
DlgDirSelect	Retrieves the current selection from a list box.
DlgDirSelectComboBox	Retrieves the current selection from the list box of a combo box.
GetDlgItemInt	Translates the text of a control in the given dialog box to an integer value.
GetDlgItemText	Retrieves the caption or text associated with a control.
GetNextDlgGroupItem	Searches for the next (or previous) control within a group of controls.
GetNextDlgTabItem	Retrieves the first control with the WS_TABSTOP style that follows (or precedes) the specified control.
IsDlgButtonChecked	Determines whether a button control is checked.
IsDialogMessage	Determines whether the given message is intended for the modeless dialog box and, if so, processes it.
SendDlgItemMessage	Sends a message to the specified control.
SetDlgItemInt	Sets the text of a control to the string that represents an integer value.
SetDlgItemText	Sets the caption or text of a control in the specified dialog box.
SubclassDlgItem	Attaches a Windows control to a **CWnd** object and makes it route messages through the **CWnd**'s message map.
ExecuteDlgInit	Initiates a dialog resource.
RunModalLoop	Retrieves, translates, or dispatches messages for a window that is in modal status.
ContinueModal	Continues a window's modal status.
EndModalLoop	Ends a window's modal status.

Data-Binding Functions

BindDefaultProperty	Binds the calling object's default simple bound

	property, as marked in the type library, to a cursor associated with a data-source control.
BindProperty	Binds a cursour-bound property on a data-bound control to a data-source control and registers that relationship with the MFC binding manager.
GetDSCCursor	Retrieves a pointer to the underlying cursor that is defined by the DataSource, UserName, Password, and SQL properties of a data-source control.

Menu Functions

GetMenu	Retrieves a pointer to the specified menu.
SetMenu	Sets the menu to the specified menu.
DrawMenuBar	Redraws the menu bar.
GetSystemMenu	Allows the application to access the Control menu for copying and modification.
HiliteMenuItem	Highlights or removes the highlighting from a top-level (menu-bar) menu item.

ToolTip Functions

EnableToolTips	Enables the tooltip control.
CancelToolTips	Disables the tooltip control.
FilterToolTipMessage	Retrieves the title or text associated with a control in a dialog box.
OnToolHitTest	Detemines whether a point is in the bounding rectangle of the specified tool and retrieves information about the tool.

Timer Functions

SetTimer	Installs a system timer that sends a WM_TIMER message when triggered.
KillTimer	Kills a system timer.

Alert Functions

FlashWindow	Flashes the window once.

MessageBox	Creates and displays a window that contains an application-supplied message and caption.

Window Message Functions

GetCurrentMessage	Returns a pointer to the message this window is currently processing. Should only be called when in an **On***Message* message-handler member function.
Default	Calls the default window procedure, which provides default processing for any window messages that an application does not process.
PreTranslateMessage	Used by **CWinApp** to filter window messages before they are dispatched to the **TranslateMessage** and **DispatchMessage** Windows functions.
SendMessage	Sends a message to the **CWnd** object and does not return until it has processed the message.
PostMessage	Places a message in the application queue, then returns without waiting for the window to process the message.
SendNotifyMessage	Sends the specified message to the window and returns as soon as possible, depending on whether the calling thread created the window.

Clipboard Functions

ChangeClipboardChain	Removes **CWnd** from the chain of Clipboard viewers.
SetClipboardViewer	Adds **CWnd** to the chain of windows that are notified whenever the contents of the Clipboard are changed.
OpenClipboard	Opens the Clipboard. Other applications will not be able to modify the Clipboard until the Windows CloseClipboard function is called.
GetClipboardOwner	Retrieves a pointer to the current owner of the Clipboard.
GetOpenClipboardWindow	Retrieves a pointer to the window that currently has the Clipboard open.
GetClipboardViewer	Retrieves a pointer to the first window in the chain of Clipboard viewers.

OLE Controls

SetProperty	Sets an OLE control property.
OnAmbientProperty	Implement ambient property values.
GetControlUnknown	Retrieves a pointer to an unknown OLE control.
GetProperty	Retrieves an OLE control property.
InvokeHelper	Invokes an OLE control method or property.

Overridables

WindowProc	Provides a window procedure for a **CWnd**. The default dispatches messages through the message map.
DefWindowProc	Calls the default window procedure, which provides default processing for any window messages that an application does not process.
PostNcDestroy	This virtual function is called by the default OnNcDestroy function after the window has been destroyed.
OnNotify	Called by the framework to inform a parent window an event has occurred in one of its controls or that the control needs information.
OnChildNotify	Called by a parent window to give a notifying control a chance to respond to a control notification.
DoDataExchange	For dialog data exchange and validation. Called by **UpdateData**.

Initialization Message Handlers

OnInitMenu	Called when a menu is about to become active.
OnInitMenuPopup	Called when a pop-up menu is about to become active.

System Message Handlers

OnSysChar	Called when a keystroke translates to a system character.
OnSysCommand	Called when the user selects a command from the Control menu, or when the user selects the Maximize or Minimize button.
OnSysDeadChar	Called when a keystroke translates to a system dead

	character (such as accent characters).
OnSysKeyDown	Called when the user holds down the ALT key and then presses another key.
OnSysKeyUp	Called when the user releases a key that was pressed while the ALT key was held down.
OnCompacting	Called when Windows detects that system memory is low.
OnDevModeChange	Called for all top-level windows when the user changes device-mode settings.
OnFontChange	Called when the pool of font resources changes.
OnPaletteIsChanging	Informs other applications when an application is going to realize its logical palette.
OnPaletteChanged	Called to allow windows that use a color palette to realize their logical palettes and update their client areas.
OnSysColorChange	Called for all top-level windows when a change is made in the system color setting.
OnWindowPosChanging	Called when the size, position, or Z-order is about to change as a result of a call to SetWindowPos or another window-management function.
OnWindowPosChanged	Called when the size, position, or Z-order has changed as a result of a call to SetWindowPos or another window-management function.
OnDropFiles	Called when the user releases the left mouse button over a window that has registered itself as the recipient of dropped files.
OnSpoolerStatus	Called from Print Manager whenever a job is added to or removed from the Print Manager queue.
OnTimeChange	Called for all top-level windows after the system time changes.
OnWinIniChange	Called for all top-level windows after the Windows initialization file, WIN.INI, is changed.

General Message Handlers

OnCommand	Called when the user selects a command.
OnActivate	Called when **CWnd** is being activated or deactivated.
OnActivateApp	Called when the application is about to be activated or deactivated.

OnCancelMode	Called to allow **CWnd** to cancel any internal modes, such as mouse capture.
OnChildActivate	Called for multiple document interface (MDI) child windows whenever the size or position of **CWnd** changes or **CWnd** is activated.
OnClose	Called as a signal that **CWnd** should be closed.
OnCopyData	Copies data from one application to another.
OnCreate	Called as a part of window creation.
OnCtlColor	Called if **CWnd** is the parent of a control when the control is about to be drawn.
OnDestroy	Called when **CWnd** is being destroyed.
OnEnable	Called when **CWnd** is enabled or disabled.
OnEndSession	Called when the session is ending.
OnEnterIdle	Called to inform an application's main window procedure that a modal dialog box or a menu is entering an idle state.
OnEraseBkgnd	Called when the window background needs erasing.
OnGetMinMaxInfo	Called whenever Windows needs to know the maximized position or dimensions, or the minimum or maximum tracking size.
OnIconEraseBkgnd	Called when **CWnd** is minimized (iconic) and the background of the icon must be filled before painting the icon.
OnKillFocus	Called immediately before **CWnd** loses the input focus.
OnMenuChar	Called when the user presses a menu mnemonic character that doesn't match any of the predefined mnemonics in the current menu.
OnMenuSelect	Called when the user selects a menu item.
OnMove	Called after the position of the **CWnd** has been changed.
OnMoving	Indicates that a user is moving a **CWnd** object.
OnDeviceChange	Notifies an application or device driver of a change to the hardware configuration of a device or the computer.
OnStyleChanged	Indicates that the ::SetWindowLong Windows function has changed one or more of the window's styles.

OnStyleChanging	Indicates that the ::SetWindowLong Windows function is about to change one or more of the window's styles.
OnPaint	Called to repaint a portion of the window.
OnParentNotify	Called when a child window is created or destroyed, or when the user clicks a mouse button while the cursor is over the child window.
OnQueryDragIcon	Called when a minimized (iconic) **CWnd** is about to be dragged by the user.
OnQueryEndSession	Called when the user chooses to end the Windows session.
OnQueryNewPalette	Informs **CWnd** that it is about to receive the input focus.
OnQueryOpen	Called when **CWnd** is an icon and the user requests that the icon be opened.
OnSetFocus	Called after **CWnd** gains the input focus.
OnShowWindow	Called when **CWnd** is to be hidden or shown.
OnSize	Called after the size of **CWnd** has changed.
OnSizing	Indicates that the user is resizing the rectangle.
OnStyleChanged	Indicates that one or more of the window's styles has changed.
OnStyleChanging	Indicates that one or more of the window's styles is about to change.

Control Message Handlers

OnCharToItem	Called by a child list box with the LBS_WANTKEYBOARDINPUT style in response to a WM_CHAR message.
OnCompareItem	Called to determine the relative position of a new item in a child sorted owner-draw combo box or list box.
OnDeleteItem	Called when an owner-draw child list box or combo box is destroyed or when items are removed from the control.
OnDrawItem	Called when a visual aspect of an owner-draw child button control, combo-box control, list-box control, or menu needs to be drawn.
OnDSCNotify	Called in response to an event that a data-source control fires when a control to which the data-source

	control is bound modifies or attempts to modify the underlying cursor.
OnGetDlgCode	Called for a control so the control can process arrow-key and TAB-key input itself.
OnMeasureItem	Called for an owner-draw child combo box, list box, or menu item when the control is created. **CWnd** informs Windows of the dimensions of the control.
SendChildNotifyLastMsg	Provides a notification message to a child window, from the parent window, so the child window can handle a task.
ReflectChildNotify	Helper function which reflects a message to its source.
OnWndMsg	Indicates if a windows message was handled.
ReflectLastMsg	Reflects the last message to the child window.
OnVKeyToItem	Called by a list box owned by **CWnd** in response to a WM_KEYDOWN message.

Input Message Handlers

OnChar	Called when a keystroke translates to a nonsystem character.
OnDeadChar	Called when a keystroke translates to a nonsystem dead character (such as accent characters).
OnHScroll	Called when the user clicks the horizontal scroll bar of **CWnd**.
OnKeyDown	Called when a nonsystem key is pressed.
OnKeyUp	Called when a nonsystem key is released.
OnLButtonDblClk	Called when the user double-clicks the left mouse button.
OnLButtonDown	Called when the user presses the left mouse button.
OnLButtonUp	Called when the user releases the left mouse button.
OnMButtonDblClk	Called when the user double-clicks the middle mouse button.
OnMButtonDown	Called when the user presses the middle mouse button.
OnMButtonUp	Called when the user releases the middle mouse button.
OnMouseActivate	Called when the cursor is in an inactive window and the user presses a mouse button.

OnMouseMove	Called when the mouse cursor moves.
OnMouseWheel	Called when a user rotates the mouse wheel. Uses Windows NT 4.0 message handling.
OnRegisteredMouseWheel	Called when a user rotates the mouse wheel. Uses Windows 95 and Windows NT 3.51 message-handling.
OnRButtonDblClk	Called when the user double-clicks the right mouse button.
OnRButtonDown	Called when the user presses the right mouse button.
OnRButtonUp	Called when the user releases the right mouse button.
OnSetCursor	Called if mouse input is not captured and the mouse causes cursor movement within a window.
OnTimer	Called after each interval specified in SetTimer.
OnVScroll	Called when the user clicks the window's vertical scroll bar.
OnCaptureChanged	Sends a message to the window that is losing the mouse capture.

Nonclient-Area Message Handlers

OnNcActivate	Called when the nonclient area needs to be changed to indicate an active or inactive state.
OnNcCalcSize	Called when the size and position of the client area need to be calculated.
OnNcCreate	Called prior to OnCreate when the nonclient area is being created.
OnNcDestroy	Called when the nonclient area is being destroyed.
OnNcHitTest	Called by Windows every time the mouse is moved if **CWnd** contains the cursor or has captured mouse input with **SetCapture**.
OnNcLButtonDblClk	Called when the user double-clicks the left mouse button while the cursor is within a nonclient area of **CWnd**.
OnNcLButtonDown	Called when the user presses the left mouse button while the cursor is within a nonclient area of **CWnd**.
OnNcLButtonUp	Called when the user releases the left mouse button while the cursor is within a nonclient area of **CWnd**.

OnNcMButtonDblClk	Called when the user double-clicks the middle mouse button while the cursor is within a nonclient area of **CWnd**.
OnNcMButtonDown	Called when the user presses the middle mouse button while the cursor is within a nonclient area of **CWnd**.
OnNcMButtonUp	Called when the user releases the middle mouse button while the cursor is within a nonclient area of **CWnd**.
OnNcMouseMove	Called when the cursor is moved within a nonclient area of **CWnd**.
OnNcPaint	Called when the nonclient area needs painting.
OnNcRButtonDblClk	Called when the user double-clicks the right mouse button while the cursor is within a nonclient area of **CWnd**.
OnNcRButtonDown	Called when the user presses the right mouse button while the cursor is within a nonclient area of **CWnd**.
OnNcRButtonUp	Called when the user releases the right mouse button while the cursor is within a nonclient area of **CWnd**.

MDI Message Handlers

OnMDIActivate	Called when an MDI child window is activated or deactivated.

Clipboard Message Handlers

OnAskCbFormatName	Called by a Clipboard viewer application when a Clipboard owner will display the Clipboard contents.
OnChangeCbChain	Notifies that a specified window is being removed from the chain.
OnDestroyClipboard	Called when the Clipboard is emptied through a call to the Windows EmptyClipboard function.
OnDrawClipboard	Called when the contents of the change.
OnHScrollClipboard	Called when a Clipboard owner should scroll the Clipboard image, invalidate the appropriate section, and update the scroll-bar values.

OnPaintClipboard	Called when the client area of the Clipboard viewer needs repainting.
OnRenderAllFormats	Called when the owner application is being destroyed and needs to render all its formats.
OnRenderFormat	Called for the Clipboard owner when a particular format with delayed rendering needs to be rendered.
OnSizeClipboard	Called when the size of the client area of the Clipboard-viewer window has changed.
OnVScrollClipboard	Called when the owner should scroll the Clipboard image, invalidate the appropriate section, and update the scroll-bar values.

Menu Loop Notification

OnEnterMenuLoop	Called when a menu modal loop has been entered.
OnExitMenuLoop	Called when a menu modal loop has been exited.

CWnd Overview | Base Class Members | Hierarchy Chart