

Create a toolbar

```
CToolBar m_wndToolBar;
...
m_wndToolBar.Create(this);

m_wndToolBar.Create(this,
    WS_CHILD | WS_VISIBLE | CBRS_BOTTOM | CBRS_FLYBY |
    CBRS_TOOLTIPS, toolbarID);

CBRS_TOP      for top location
CBRS_FLYBY   for display in status bars
toolbarID    to create from toolbar resource.
```

Design the toolbar with the resource editor or use default AFX_IDW_TOOLBAR

double click toolbar button in resource editor to add tooltip or flyby text
(with status bar flyby is shown rather than tooltip)

Load the toolbar resource after creation

```
m_wndToolBar.LoadToolBar(IDR_TOOLBAR1);
```

There are many styles you can set for your toolbar buttons

```
TBBS_CHECKBOX      check push button
TBBS_CHECKGROUP    radio group
TBBS_BUTTON        standard push button

// set the third button in toolbar
m_wndToolBar.SetButtonSyle(2, TBBS_CHECKBOX);
```

Respond to toolbar buttons with a command range for buttons in an
OnButton(UINT nID) method.

Toolbars can be docking and floating

You need to EnableDocking for the toolbar (with CControlBar base
class for toolbar) and frame docking is enabled with an alignment
arguments

```
CBRS_ALIGN_*      LEFT RIGHT TOP BOTTOM ANY
```

```
CStatusBar m_wndStatusBar;  
UINT nIndicator = ID_SEPARATOR; // a null resource  
...  
m_wndStatusBar.Create(this);  
m_wndStatusBar.SetIndicators(&nIndicator, 1);
```

Initialize the status bar with SetIndicators

```
BOOL SetIndicators( const UINT* lpIDArray, int nIDCount );  
    array of resources (strings for each pane in status bar  
    count of panes
```

four MFC defined indicators

```
ID_INDICATOR_* CAPS NUM SCRL KANA
```

status bar places indicators in order. The first is left justified, the rest are right justified and the bar fills horizontally stretching the left indicator.

Setting values in panes

Setting text values

```
m_wndStatusBar.SetPaneText(0, msg, TRUE);
```

sets pane 0 to have the string msg

A string resource ID for an indicator can have a toggled display by enabling the pane. When enabled the string is displayed, else blank.

CMainFrame's message map

```
ON_UPDATE_COMMAND_UI (ID_INDICATOR, OnUpdateIndicator)
```

Method definition

```
void CMainFrame::OnUpdateIndicator(CCmdUI * pCmdUI) {  
    pCmdUI -> Enable(m_boolean); }  
}
```

ON_UPDATE_COMMAND_UI is called when application is idle. Thus status variable values can be set, and display is automatically updated. Default implementation OnIdle()

Create windows with WS_VSCROLL and/or WS_HSCROLL

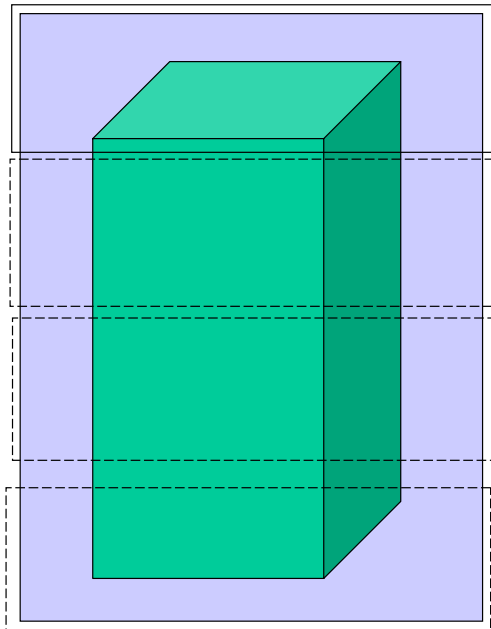
```
Create(NULL, _T ("Scroll Window"),
        WS_OVERLAPPEDWINDOW | WSVSCROLL)
```

Set information about scrolling in teh SCROLLINFO struct contains:

size, mask, type of scrollbar, min and max, pos etc. (see pg 83)

masks: SIF_PAGE	nPage holds page size
SIF_POS	nPos holds scroll bar position
SIF_RANGE	nMin and nMax == range
SIF_ALL	combines above 3

```
SCROLLINFO si;
si.fMask = SIF_ALL;
si.nMin = 0;        si.nMax = 99;
si.nPage = 25;      si.nPos = 0;
SetScrollInfo(sb_vert, &si, TRUE);
```



client window is 100 high
 scroll page size is 100
 workspace are is 400 high
 scroll range should be 0..399
 scrollbar nMax should be 300

```
SCROLLBAR si;
...
si.fMask = SIF_ALL;
si.nMin = 0;
si.nMax = 399;
si.nPos = 0;
si.nPage = 100;
```

Processing Scroll Bar Messages

```
afx_msg void OnVScroll (UINT nCode, UINT nPos,
    CScrollBar * pScrollBar);

nCode type of event
    SB_LINEUP, SB_LINEDOWN, // up down arrows
    SB_PAGEUP, SB_PAGEDOWN, // between thumb & arrow
    SB_ENDSCROLL,          // after any sb events
    SB_THUMBTRACK,         // thumb dragged
    SB_THUMBPOSITION       // thumb released

nPos latest thumb position
```

Need to determine the position of the top of the viewable window inside the OnVScroll method. (vOffset)

This value can then be used to set the window's origin in OnPaint

```
dc.SetWindowOrg(0, vOffset);
```

Last OnVScroll should set scrollbar position and scroll the window

```
CWnd::SetScrollPos(int nBar, int nPos,
    BOOL bRedraw = TRUE);
```

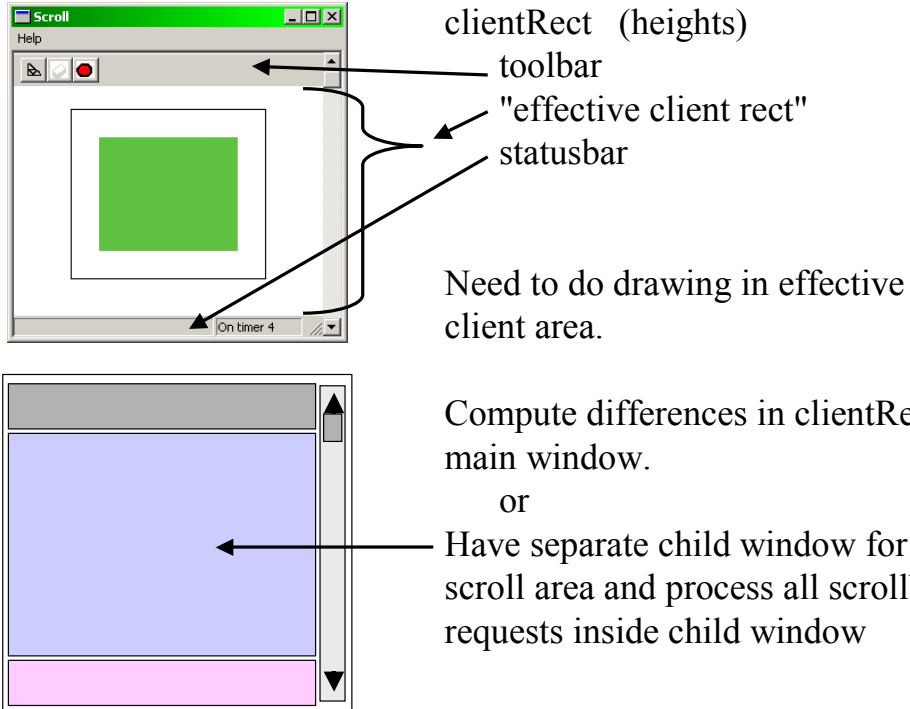
CWnd::ScrollWindow scroll the window

```
ScrollWindow(int xAmount, int yAmount,
    LPCRECT lpRect = NULL, LPCRECT lpClipRect = NULL)

ScrollWindow(0, vOffset); // negative to scroll left up

ScrollWindow(100, 100);
    scrolls the entire window right && down 100 pixels.
```

After repainting window to reflect scroll position changed use SetScrollInfo to reset the scrollbar position and thumb location.

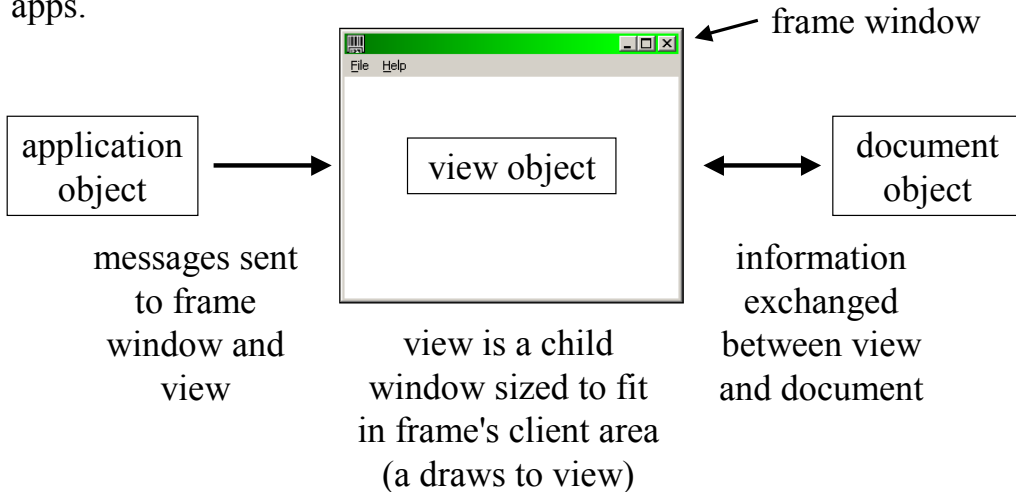


Document View

A framework -- generate using AppWizard build tools

Scrolling window is a "view" on a larger "model".

MFC Document View architecture better suited for scrolling window apps.



```

    ...
    CSingleDocTemplate* pDocTemplate;
    pDocTemplate = new CSingleDocTemplate(
        IDR_MAINFRAME,          // dynamic construction
        RUNTIME_CLASS(CAppNameDoc), // document object
        RUNTIME_CLASS(CMainFrame), // main SDI frame window
        RUNTIME_CLASS(CAppNameView)); // view object
    AddDocTemplate(pDocTemplate); // doc list SDI 1 MDI 1+
    ...
    CCommandLineInfo cmdInfo;
    ParseCommandLine(cmdInfo); // Parse command line args

    // Dispatch commands specified on the command line
    if (!ProcessShellCommand(cmdInfo))
        return FALSE;

    // The one and only window has been initialized, so
    // show and update it.
    m_pMainWnd->ShowWindow(SW_SHOW); // show frame window
    m_pMainWnd->UpdateWindow();
    return TRUE; }

```

All data (*m_**variables) relevant to application are in the document object.

Document can access its associated view(s)

```

POSITION p = GetFirstViewPosition();
CView * pView = GetNextView(pos);

```

All drawing is done to view. (MainFrame clientRect is covered).

View(s) can access its associated document

```

CApDoc * pDoc = GetDocument();

```

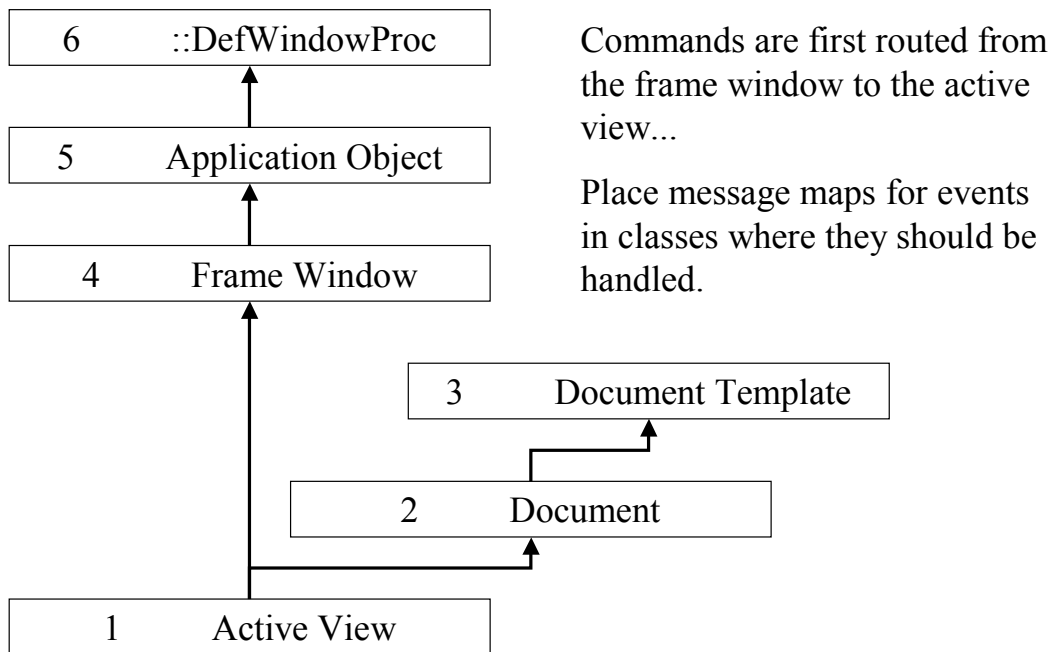
In document/view framework WM_PAINT is mapped to OnDraw(CDC *) -- there is no need to get ClientDCs.

```

CApView::OnDraw(CDC * pDC) {
    ...
}

```

Events && Doc/View



CScrollView

Derive view class from CScrollView

WM_VSCROLL, WM_HSCROLL bars shown as needed

Implement OnDraw

window scrolls, thumb is update (pos, size) (default size 100 pixels)

Optionally override OnInitialUpdate() to call SetScrollSizes for dimensions. 4 args:

int mapping mode for unit of dimensions

SIZE or Csize for logical dimensions

SIZE or CSize for page size (scroll w/ shaft clicks) optional

SIZE or CSize for line size (scroll w/ arrow clicks) optional

```
void CAppNameView::OnInitialUpdate() {
    SetScrollSizes(MM_TEXT, CSize(800, 600)); }

```

Other functions:

```
CPoint pos = GetScrollPosition();
ScrollToPosition(CPoint(100, 100));

```

CHTMLView displays html documents

provide URL displays like Internet Explorer

Active Document container -- will display documents created with MS Word, Excel, others, contents on disk

CListView displays list of items or text or images (flat data)

presentation modes: report, large icon, small icon, list

report displays text only subitems

selection modes: single, multiple

items are sortable

CTreeView displays hierarchical items of text or images in tree structure like Windows Explorer

expand or collapse display to show or hide subitems

drag and drop placement

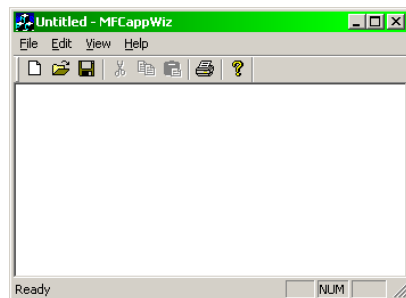
in place editing

Custom views....

Using frameworks, application builders, and code generators increase productivity.

Modern GUI environments have Integrated Design and Development Environments that use an application framework architecture.

- initial framework builder
 - code generation
 - file i/o
- resource editors
- class editors
- "controls", forms, dialogs
- class, resource, help browsers
- integrated edit, compile, debug



GUI development environments have initial steep learning curve that can provide productivity through code generation and error reduction.

< generate MFCAppWiz example and examine files.>