ObjectWindows Tutorial Examples

The ObjectWindows tutorial examples show how to create a drawing application from the ground up. For complete documentation, see the ObjectWindows Tutorial.

Project file

STEPS.IDE

Header files

STEP16.H

STEP16DV.H

STEP17.H

STEP17DV.H

STEP18.H

STEP18DV.H

Source files

STEP01.CPP

STEP02.CPP

STEP03.CPP

STEP04.CPP

STEP05.CPP

STEP06.CPP

STEP07.CPP

STEP08.CPP

STEP09.CPP

STEP10.CPP

STEP11.CPP

STEP11DV.CPP

STEP12.CPP

STEP12DV.CPP

STEP13.CPP

STEP13DV.CPP

STEP14.CPP

STEP14DV.CPP

STEP15.CPP

STEP15DV.CPP

CNTRLLER.CPP

STEP16.CPP

STEP16DV.CPP

CNTRL16.CPP

STEP17.CPP

STEP17DV.CPP

CNTRL17.CPP

STEP18.CPP

STEP18DV.CPP

CNTRL18.CPP

ObjectWindows OLE Tutorial Examples

The ObjectWindows OLE tutorial examples show how to create a simple OLE automation server and container.

Project file

OWLOLE.IDE

Source files

OWLOLE1.CPP

OWLOLE2.CPP

OWLOLE3.CPP

SYSMETRI Example

Project file

SYSMETRI.IDE

DRAWSTAT Example

Project file

DRAWSTAT.IDE

DRAWEDGE Example

Project file

DRAWEDGE.IDE

TBEXPERT Example

Project file

TBEXPERT.IDE

TTT (OLE) Example

Demonstrates a Tic-Tac-Toe OLE server.

Project file

TTT.IDE

OCXDLG Example

Shows how to use OCX controls in a dialog box.

Note: In order to run this example you must first install the Visual Components OCX controls Formula One Spread Sheet and First Impression Chart.

Project file

OCXDLG.IDE

CHESS Example

Owl Chess is based upon the chess program originally released with Turbo Pascal GameWorks.   The chess engine was ported over to C, and is essentially unchanged here. This example demonstrates how OWL may be used to place a Windows wrapper around DOS code, whether it is written in C or C++.

Project file

CHESS.IDE

WINSOCK Example

This Winsock example is an application that demonstrates some features of the OWL Winsock classes. It uses only asynchronous (non-blocking) Winsock calls and uses socket 'external' notification rather than internal notification. External notification is the way most Winsock apps do FD_XXX notifications; see the documentation for the Winsock classes for more information.

Project file

WINSOCK.IDE

VERINFO Example

Shows how to use the TModuleVersionInfo class.

Project file

VERINFO.IDE

Source files

VERINFO.CPP

TOOLBRES Example

Shows how to use a toolbar resource.

Project file

TOOLBRES.IDE

Source files

TOOLBRES.CPP

TOOLBOX Example

Shows how to use the TToolBox class.

Source files

TOOLBOXX.CPP

SHELL Example

Project file

SHELL.IDE

ROLLDIAL Example

Shows how to use the TRollDialog class.

Project file

ROLLDIAL.IDE

Source files

ROLLDLGX.CPP

PICTWIND Example

Shows how to use the TPictureWindow class.

Project file

PICTWIND.IDE

Source files

PICTWINX.CPP

CTXTHELP Example

Shows how to use the THelpFileManager class to add context-sensitive Help to an application.

Project file

CTXTHELP.IDE

Source files

CTXTHELP.CPP

CHECKLST Example

Shows how to use the TCheckList class.

Project file

CHECKLSX.IDE

Source files

CHECKLSX.CPP

LED Example

Shows how to create an LED-like display.

Project file

LED.IDE

Header files

LEDWIND.H

Source files

LED.CPP

LEDWIND.CPP

MDIOLE Example

An MDI application that uses ObjectComponents classes to support OLE in its documents.

Project file

MDIOLE.IDE

Header files

MDIOLE.H

SAMPCONT.H

Source files

MDIOLE.CPP

SAMPCONT.CPP

SDIOLE Example

An SDI application that uses ObjectComponents classes to support OLE.

Project file

SDIOLE.IDE

Header files

SAMPCONT.H

SDIOLE.H

Source files

SAMPCONT.CPP

SDIOLE.CPP

OUTDEB32 Example

Project file

OUTDEB32.IDE

Source files

OUTDEB32.CPP

DIAGXPRT Example

DIAGXPRT displays diagnostic messages sent by the OutputDebugString() function or by the RTL diagnostic macros TRACEX and WARNX. OWL diagnostics can be enabled or disabled by using the Configure button.

There are two levels of diagnostics used in OWL: 0 and 1. You can set the levels for each area. You can also add your own diagnostic groups. Each new group will be written to the OWL.INI file and the declaration placed on the Clipboard for inclusion in your code.

The OWL.INI file is only read when your application starts. Please see the RTL documentation for a complete description of the diagnostic macros and their uses. You can also find additional examples in the OWL source directory.

Project file

DIAGXPRT.IDE

Header files

DIAGXPRT.H

SETUP.H

SERIALZE.H

Source files

DIAGXPRT.CPP

CBACK.CPP

SETUP.CPP

TOOLHELP.CPP

SERIALZE.CPP

DRAGDROP Example

Demonstrates how to handle the dragging and dropping of files into the client area of the main window under Windows 3.1.

The TMyWindow class maintains the drag-and-drop information in a list of Lists. Each sub-List is a set of files that were dropped.

Comments in the code give step-by-step instructions on how to build a drag-and-drop application, describe how to avoid common pitfalls, and emphasize important lines of code that affect the performance of your application.

Project file

DRAGDROP.IDE

Source files

DRAGDROP.CPP

HELP Example

An application that implements context-sensitive Help for menu choices. To use the application, press F1 when a menu item is highlighted. The program checks for F1 being down in the WM_ENTERIDLE event-handler. If the F1 key is down, the application sets a flag and simulates the selection of the menu item. The online Help is then shown in the command message for that menu item. When the command is received, we just check to see if the flag has been set, indicating that the user wants help on the command.

Project file

HELP.IDE

Source files

HELP.CPP

MCISOUND Example

This example demonstrates the use of MCI APIs in a Windows 3.1 OWL application. You must have a sound card, speaker, and a corresponding device driver installed under Windows 3.1 in order for this example to work.

1. Use the sound applet in Windows Control Pannel to generate a sound. You may copy one of the .WAV files from the WINDOWS subdirectory in your system to this example's subdirectory.
2. Run the .EXE.
3. Choose File|Open and select a .WAV file.
4. Choose Control|Play to play the waveform.

The Control menu lets you stop/play/pause and resume. The scrollbar allows random access through the waveform while it is playing.

Project file

MCISOUND.IDE

Header files

MCISOUND.H

Source files

MCISOUND.CPP

PROGMAN Example

PROGMAN engages in a DDE conversation with the Program Manager to create program groups with a user-specified list of program items.

Project file

PROGMAN.IDE

Header files

PROGMANX.H

Source files

PROGMANX.CPP

SYSINFO Example

Project file

SYSINFO.IDE

Header files

SYSINFO.H

Source files

SYSINFO.CPP

TRUETYPE Example

Shows how to display and print TrueType fonts.

Project file

TRUETYPE.IDE

Header files

TRUETYPE.H

Source files

TRUETYPE.CPP

MCIWND Example

Header files

MCIWND.H

MCIWNDX.H

Source files

MCIWND.CPP

MCIWNDX.CPP

ACLOCK Example

Demonstrates using the system clock to display the current time. Also shows how to use animation and sound effects.

Project file

ACLOCK.IDE

Header files

ACLOCK.H

Source files

ACLOCK.CPP

GDIDEMO Example

The GDIDEMO example shows how to display graphics of several formats.

Project file

GDIDEMO.IDE

Header files

ARTYPRIV.H

BITBLT.H

ARTY.H

LINE.H

DEMOBASE.H

FONTX.H

Source files

ARTY.CPP

LINE.CPP

GDIDEMO.CPP

FONTX.CPP

BITBLT.CPP

CALC Example

CALC is a calculator created with ObjectWindows.

Project file

CALC.IDE

Source files

CALC.CPP

CURSOR Example

CURSOR shows how to change the cursor and track its position.

Project file

CURSOR.IDE

Header files

CURSOR.H

Source files

CURSOR.CPP

BMPVIEW Example

BMPVIEW shows Dibs, bitmaps and palettes in a scrolling window. Also uses diagnostics to trace through some routines.

Project file

BMPVIEW.IDE

Header files

BMPVIEW.H

Source files

BMPVIEW.CPP

PAINT Example

PAINT is a drawing program created with ObjectWindows.

Project file

PAINT.IDE

Header files

DIBATTR.H

Source files

PAINT.CPP

DIBATTR.CPP

OWLCMD Example

This program demonstrates how to use th eTFloatingFrame class. It uses a class derived from TComboBox that catches the Return key and executes the command that is typed. If the command executes sucessfully it is stored in the combo box for future retrieval.

Commands supported

DOS	Executes command.com
DIR	Launches WINFILE
CD x$	Changes default directory (CD\ CD \)
EXIT	Closes OWLCMD
x$:	Changes default drive (D:)
x$	Executes command x$

Project file

OWLCMD.IDE

Source files

OWLCMD.CPP

SCRNSAVE Example

The SCRNSAVE example shows how to create a screen saver using ObjectWindows.

Project file

OWLSCRN.IDE

Header files

OWLSCRN.H

TSCRNSAV.H

Source files

OWLSCRN.CPP

TSCRNSAV.CPP

DRAW Example

DRAW shows how to work with metafiles.

Project file

DRAW.IDE

Source files

DRAW.CPP

MTHREAD Example

MTHREAD shows how to create a multi-threaded drawing application.

Project file

MTHREAD.IDE

Header files

LINE.H

ARTY.H

ARTYPRIV.H

DEMOBASE.H

Source files

MTHREAD.CPP

LINE.CPP

ARTY.CPP

MDIFILE Example

MDIFILE shows how to create a multiple-document file editor.

Project file

MDIFILE.IDE

Header files

MDIFILE.H

NEWDEL.H

Source files

MDIFILE.CPP

NEWDEL.CPP

HELLO Example

HELLO shows the most basic OWL application.

Project file

HELLOAPP.IDE

Source files

HELLOAPP.CPP

SDIFILE Example

SDIFILE shows how to create a file editor.

Project file

SDIFILE.IDE

Source files

SDIFILE.CPP

PEEPER Example

Allows you to "peep" at a window, displaying information such as:

- the class of the window (e.g. Edit, ListBox, etc.)
- the dimensions of the window (left, top, right, bottom)
- whether or not the window has a menu
- whether or not the window has children

To use the application,

1. Press 'Peep!' button to select windows.
2. While peeping press left mouse button to select window under cursor.
3. When finished peeping press right mouse button to return to command mode.

Project file

PEEPER.IDE

Header files

PEEPER.H

Source files

PEEPER.CPP

FILEBROW Example

FILEBROW shows how to create a file browser.

Project file

FILEBROW.IDE

Source files

FILEBROW.CPP

NOTEBOOK Example

The NOTEBOOK example shows how to create an application with right-, left-, top-, and bottom-aligned tabs similar to the notebook tabs found in many popular Windows applications.

Project file

NOTEBOOK.IDE

Header files

NOTEBOOK.H

UTILS.H

VSCROLL.H

HSCROLL.H

NB.H

Source files

VSCROLL.CPP

UTILS.CPP

NOTEBOOK.CPP

HSCROLL.CPP

NB.CPP

APPLAUNC Example

APPLAUNC shows how to launch other applications from within an application.

Project file

APPLAUNC.IDE

Header files

APPWIN.H

APPMGR.H

APPBTNBA.H

DIALOGS.H

APPBTN.H

Source files

APPLAUNC.CPP

APPBTN.CPP

APPWIN.CPP

DIALOGS.CPP

APPBTNBA.CPP

EXITWIN.CPP

APPMGR.CPP

INTLDEMO Example

Demonstrates much of the functionality afforded by the SetLocale function and the locale libraries. Also demonstrates programming a user interface that is language-independent and can change languages at run-time.

Project file

INTLDEMO.IDE

Header files

INTLDEMO.H

Source files

INTLDEMO.CPP

FMTVAL.CPP

BLAZER Example

The BLAZER sample application illustrates how to use two of the of the common controls: TTreeWindow and TListWindow. TTreeWindow, which is based on the TreeView common control, displays information in a hierarchical format. TListWindow, which is based on the ListView common control, displays information in either list, small icon, large icon, or detailed format.

Project file

BLAZER.IDE

Header files

BLAZER.H

PROPDLGS.H

Source files

BLAZER.CPP

ABOUT.CPP

CLIENT.CPP

PROPDLGS.CPP

BLAKJACK Example

This is a subset of a standard blackjack game. It uses a card VBX control to display the cards.

Objective

One player and one dealer can play this game.

The player enters amount of money using the "Bankroll" button at the begining of the game. After entering the bankroll amount, the player can go on pressing "Hit" button until the score is near 21. If the player scores more than 21, he looses. The trick is to hit "Stand" button when the score is near 21. After you loose or win, you can bet again using the "Bet" button. Player plays the game until the bankroll is exhausted, at that time he can input more money in bankroll.

Design Overview

First the Bankroll is entered by the user and stored in the "Bankroll" class. The increment and decrement of the bankroll is done by the member funtions in that class.

52 cards are "new"-ed of type "TVbxMhCardDeck" and stored in the array "TBlackjack::ppVBXCard]" in the constructor of "TBlackjack" class. "TVbxMhCardDeck" type of cards are VBX controls. The "Card" object stores only the Suit and Number information. ("Card" object is defined in blakjack.h)

When a "Card" is displayed the Suit and Number informations are taken from the "Card" object, the displayable VBX card is taken from "TBlackjack::ppVBXCardVBXCardCount]" array. Each VBX card can have 52 possible values. The VBX card is displayed according to the above Suit and Number information.

"TBlackjack::VBXCardCount", points to the next VBX card in the "TBlackjack::ppVBXCard]" array which is available. eg: Count 12 means, cards from ppVBXCard0] through ppVBXCard11] have already been dealt and displayed, and ppVBXCard12] is the next VBX card available.

Suffling and dealing are done using "Card" and "Deck" objects. "TBlackjack::ppVBXCard]" array only holds the displayable VBX cards. Each "Card" object stores an array index of the "TBlackjack::ppVBXCard]" array in "Card::pVBXCard" data member. The VBX card at this index(in "TBlackjack::ppVBXCard]" array) is used to display that particular "Card" object. This keeps the engine and UI part seperate.

Dealer is assumed to have infinite amount of money.

The cards in a particular suit are numbered from 0 - 12 eg: Ace->0, Two->1..., Jack->10, King->11, Queen->12 These numbers have nothing to do with the actual blackjack points, it is used only to keep track of the cards.

Project file

BLAKJACK.IDE

Header files

BLAKJACK.H

OWLMAIN.H

MHCD2001.H

Source files

OWLMAIN.CPP

BLAKJACK.CPP

BLOCKS Example

Turbo Blocks is a faithful reproduction of the popular game Tetris. The game is played on a 10x20 grid. Blocks of various shapes fall from the top of the grid. They stop when they hit the bottom of the grid, or another block. Blocks can be moved left/right (using the arrow keys), or rotated (with the spacebar). If an entire row of the game grid is filled, it will disappear, and the remaining blocks will drop down to fill in the space. The object is to continue the game without letting the blocks reach the top of the grid.

The following features were not implemented, and are left as an exercise to the user:

- score
- level count
- line count
- display of the next block that will appear

Project file

BLOCKS.IDE

Source files

BLOCKS.CPP

METEOR Example

METEOR shows how to create an Asteroids-like game.

Project file

METEOR.IDE

Header files

SPRITE.H

Source files

SPRITE.CPP

METEOR.CPP

CRIBBAGE Example

How to play

Turbo Cribbage implements a 2 handed cribbage game. The game is played to 121 points, and because points are accumulated in small amounts, a pegboard is used to keep track of the scores. Each round consists of several phases: the deal, discard, cut for starter, card play, show points.

The Deal. The deal alternates between players each round. Dealer deals six cards to each player.

Discard. Each player discards 2 cards into the crib.

Cut for starter. Pone cuts the deck, and dealer turns over the top card. If the starter is a jack, dealer takes 2 points (heels).

Card play. Pone plays the first card, and announces its face value (face cards have a value of 10). Dealer then plays a card, announcing the total of the two cards. Play continues alternately, each player announcing the running total of the cards, continuing until a player is unable to play without the sum exceeding 31. At this point, the player says 'go'. The other player must continue to play cards if they can without exceeding 31. Then they take 1 or 2 points for the go (1 point if the sum is less than 31, 2 points if it equals 31). The count then begins again at zero, and play continues with the player who called 'go'. Play continues until both players are out of cards. The player who plays the last card takes 1 point if the sum is less than 31, or 2 if it equals 31.

Scoring during play. In addition to points for 'go', the following points may also be scored:

Fifteen	If the sum of the cards reaches fifteen, peg 2.
Pairs	If the card played matches the previous card, peg 2. If the last 3 cards match, peg 6 (for 3 pair). If the last 4 cards match, peg 12 (6 pair).
Runs	If the card played forms a sequence with 2 or more of the previous cards, peg 1 point for each card in the run. The run cannot be broken. For exampole, 3,5,6,2 = run of 4 cards; 4 points; 3,9,1,2 = no run; the 9 breaks the run of 1,2, and 3.

Showing points. After cards have been played, each player shows their hand, and counts the points. The starter can be used when counting points in the hand. Points are scored as follows:

2 points	each pair
2 points	each combination of cards which adds to 15
3 points	each 3 card run
4 points	each 4 card run
5 points	each 5 card run
1 point	nobs (a jack which matches the suit of the starter)
4 points	a 4 card flush (not using the starter)
5 points	a 5-card flush (using the starter). A 5-card flush cannot be counted in the crib.

It is possible to have multiple runs, for example:

1,2,3,3	double 3-card run (2*3 + 2 for pair = 8 points)
1,2,2,3,3	quadruple 3-card run (4*3 + 4 for 2 pairs = 16 points)
1,2,2,2,3	triple 3-card run (3*3 + 6 for 3 pairs = 15 points)

If a player overlooks some points, and the opponent notices, they may call "Muggins!" and claim the missed points. Be carefull, the computer is _very_ observant.

After the hands have been counted, the cards are placed back into the deck, the deck is shuffled, and the next hand is dealt.

The game is over when one player reaches 121. The game ends immediately, and no other points are counted. This is why the order of the showing of hands is important!

Notes. Turbo Cribbage does not implement a smart computer player. The computer simply discards the first 2 cards in its hand, and plays its cards in order.

Cribbage Glossary

Term	Definition
Dealer	The player who dealt the hand
Pone	The nondealer
Starter	The upturned card on the deck
Heels	2 points for the dealer, if the starter is a jack
Nobs	When counting points in the hand, a jack which matches the suit of the starter is worth 1 point
Crib	A third hand made up of cards discarded from the players hands. The crib belongs to the dealer.
Show points	After the cards have been played, each player in turn shows their hand and counts the points. The hands are counted in strict order: pone, dealer, then dealers crib.
Muggins	If a player fails to count all the points in their hand, the opponent may call "Muggins!" and claim the overlooked points for themself

Project file

CRIBBAGE.IDE

Header files

CARDDISP.H

CARDS.H

CRIBBAGE.H

BOARD.H

DIALOGS.H

Source files

BOARD.CPP

CARDDISP.CPP

DIALOGS.CPP

CARDS.CPP

CRIBBAGE.CPP

SWAT Example

SWAT is an example of an "interactive debugging game."

Project file

SWAT.IDE

Header files

SWAT.H

Source files

SWAT.CPP

TTT Example

Plays a game of TicTacToe with the user.

Class	Description
TTTTGameApp	Main TicTacToe application, derived from TApplication
TGameWindow	Main window for the app, derived from TWindow
Square	Game squares (windows), derived from TWindow
TGameAboutBox	A TDialog box for info about TicTacToe
TGameOptionsBox	A TDialog box for setting TicTacToe options
YouMeRadioButton	A radio button which controls game settings
XORadioButton	A radio button which controls game settings

Project file

TTT.IDE

Header files

TTT.H

Source files

TTT.CPP

TTT2 Example

Plays a game of TicTacToe with the user.

Class	Description
TTTTGameApp	Main TicTacToe application, derived from TApplication
TGameView	View window for the game, derived from TToolBox
Square	Game squares (gadgets), derived from TButtonGadget
TGameAboutBox	A TDialog box for info about TicTacToe
TGameOptionsBox	A TDialog box for setting TicTacToe options
YouMeRadioButton	A radio button which controls game settings
XORadioButton	A radio button which controls game settings

Project file

TTT.IDE

Header files

TTT.H

Source files

TTT.CPP

DLLHELLO Example

DLLHELLO demonstrates how to create a simple DLL.

Project file

DLLHELLO.IDE

Header files

DLLHELLO.H

CALLDLL.H

Source files

CALLDLL.CPP

DLLHELLO.CPP

RESOURCE.CPP

INSTANCE Example

INSTANCE shows how to create an application that supports multiple instances.

Project file

INSTANCE.IDE

Source files

INSTANCE.CPP

MDIDLG Example

Header files

MDIDLG.H

TEST.H

Source files

APP.CPP

CLIENT.CPP

MDIDLG.CPP

TEST.CPP

MDISTRM Example

MDISTRM shows a persistent desktop using the MDI metaphor. It is identical to the owlapi\mdi example, but with streaming code added to implement persistence for the frame, parent (client) window, and children.

Project file

MDISTRM.IDE

Header files

MDISTRM.H

Source files

MDISTRM.CPP

MODALWIN Example

MODALWIN demonstrates how to create a modal window.

Project file

MODALWIN.IDE

Header files

MODALWIN.H

Source files

MODALWIN.CPP

NOTIFY Example

NOTIFY shows how to notify the parent window of an event that occurs on one of its controls.

Project file

NOTIFY.IDE

Header files

NOTIFY.H

Source files

NOTIFY.CPP

OWNERDRA Example

OWNERDRA shows how to create and use an owner-draw button.

Project file

OWNERDRA.IDE

Header files

OWNERDRA.H

Source files

OWNERDRA.CPP

POPUP Example

Project file

POPUP.IDE

Source files

POPUP.CPP

SIMBOR Example

SIMBOR shows how to simulate a Borland-style dialog box.

Project file

SIMBOR.IDE

Header files

SIMBOR.H

Source files

APP.CPP

BORDLG.CPP

SIMBOR.CPP

TRANSFER Example

The TRANSFER example shows how to use structures to transfer data between controls.

Project file

TRANSFER.IDE

Header files

TRANSFER.H

Source files

TRANSFER.CPP

DYNAMENU Example

DYNAMENU shows how to create dynamic menus.

Project file

DYNMNU.IDE

Header files

DYNMNU.H

Source files

APP.CPP

DMDECFR.CPP

DYNMNU.CPP

COLORDLG Example

The COLORDLG example shows how to create a custom control and a dialog box that uses it.

Project file

COLORDLG.IDE

Header files

CCTLTEST.H

COLORDLG.H

USECDLL2.H

Source files

COLORDLG.CPP

USECDLL2.CPP

CCTLTEST.CPP

TREEWIND Example

The TREEWIND example illustrates how to use OWL's TTreeWindow encapsulation of the TreeView common control.

The example is designed to be minimal. For a more complete example, please see the BLAZER OWLAPPS example.

Project file

TREEWIND.IDE

Source files

TREEWINX.CPP

GLYPHBTN Example

The GLYPHBTN example shows how to use the TGlyphButton class.

Header files

GLYPHBTN.H

Source files

GLYPHBTX.CPP

LISTWIND Example

The LISTWIND example illustrates how to use OWL's TListWindow encapsulation of the ListView common control.

The example is designed to be minimal. For a more complete example, please see the BLAZER OWLAPPS example.

Project file

LISTWIND.IDE

Source files

LISTWINX.CPP

BUTTON Example

The BUTTON sample application illustrates how to create PushButtons, CheckBoxes, RadioButtons and GroupBoxes at runtime. The sample also illustrates an enhancement provided by the ObjectWindows Library: GroupBox notifications. The following sections describe the BUTTON sample and GroupBox notifications.

Sample Overview

The BUTTON sample is a simple SDI application. The client window, derived from TWindow creates several controls and provides message handlers to handle notifications from the controls.

GroupBox Notifications

GroupBoxes provide a visual interface for grouping controls such as CheckBoxes or RadioButtons. From a programming standpoint, however, the individual controls within a groupbox communicate directly to their parent window via notification messages. This requires that a program provides handlers for each control within the GroupBox. ObjectWindows allows you to provide a single handler which is notified whenever the state of a CheckBox or RadioButton within the GroupBox changes.

If a GroupBox pointer is specified when creating a CheckBox or RadioButton object (i.e. TCheckBox or TRadioButton), the latter will invoke the 'SelectionChanged' method of the TGroupBox when its state changes. In turn, the 'SelectionChanged' method of the Groupbox sends a notification message to the parent window. This mechanism allows an application to detect changes in checkboxes or radiobuttons by simply handling the notification relayed by the GroupBox. The BUTTON sample illustrates this technique with the EV_CHILD_NOTIFY_ALL_CODES macro and the 'HandleGroupBoxMsg' handler.

Project file

BUTTON.IDE

Source files

BUTTONX.CPP

COMBOBOX Example

Shows how to use the TComboBox class.

Project file

COMBOBOX.IDE

Source files

COMBOBXX.CPP

COMMDLG Example

The COMMDLG example show how to use the Common Dialog classes.

The main window has menu selections for opening a file, changing the font, and changing the color used for the selected font. When a file is selected the name is displayed on the client area of the window.

Project file

COMMDLG.IDE

Source files

COMMDLGX.CPP

DOCVIEW Example

A minimal doc/view application. Doc/views must be linked in.

Note: this example interprets command line flags to select frame type.

Project file

DOCVIEW.IDE

Header files

ODLISTBX.H

DUMPVIEW.H

Source files

XCPTVIEW.CPP

EDITLIST.CPP

ODLISTBX.CPP

DOCVIEWX.CPP

INFOVIEW.CPP

LINEDOC.CPP

ODLISTVW.CPP

DVLOADER.CPP

DUMPVIEW.CPP

EDIT Example

Shows how to use the TEdit class.

Project file

EDIT.IDE

Source files

EDITX.CPP

EDITSEAR Example

Shows how to use the TEditSearch class.

Project file

EDITSEAR.IDE

Source files

EDITSEAX.CPP

GAUGE Example

The GAUGE example illustrates how to use the different types of gauges available with OWL. A gauge is similar to a scroll bar. It has a range and a current position.

There are three styles of gauges used in this example. The top gauge is a progress bar. It is typically used to display percentage of completion.

The second gauge, with the LED style, displays the current position in blocks. This gauge style can be a native control or it can be emulated. Your source code will not need to know the difference.

The third control in this example is a slider control. You can use the slider to manipulate the current position of the two gauges to see how they are visually different.

Project file

GAUGE.IDE

Source files

GAUGEX.CPP

GROUPBOX Example

Shows how to use the TGroupBox class.

Project file

GROUPBOX.IDE

Source files

GROUPBXX.CPP

LAYOUT Example

Interactive program to demonstrate TLayout window.   It creates a frame window, colored child windows, and a client window.

The menu choice lets you bring up a dialog for initializing a TLayoutMetrics structure for the various child windows (the dialog is modeless, so you can layout several windows at the same time). A TLayoutMetrics (declared in layoutwi.h) has four TLayoutConstraints in it, X, Y, Width and Height. When the dialog comes up, you choose which constraint you want to select, then set the various members of that constraint.

Note: there are some constraints that you don't do with layout windows. For example, if you constrain the lmWidth edges of a X constraint, it will cause an error.   (The lmWidth edge is best constrained through the Width constraint).

Project file

LAYOUT.IDE

Header files

LAYOUT.H

LAYDIA.H

Source files

LAYOUT.CPP

LAYDIA.CPP

LISTBOX Example

Shows how to use the TListBox class.

Project file

LISTBOX.IDE

Source files

LISTBOXX.CPP

MDI Example

Shows how to create a simple MDI application.

Project file

MDI.IDE

Source files

MDI.CPP

PALETTE Example

Shows how to use the TPalette class.

Project file

PALETTE.IDE

Header files

PALETTE.H

Source files

PALETTEX.CPP

PRINTING Example

The PRINTING example displays and prints a ruler using the OWL printer classes.

Project file

PRINTING.IDE

Source files

PRINTING.CPP

PRNTPREV Example

Shows how to implement a Print Preview dialog box.

Project file

PRNTPREV.IDE

Source files

PRNTPREV.CPP

SCROLLBA Example

Shows how to use the TScrollBar class.

Project file

SCROLLBA.IDE

Source files

SCROLLBX.CPP

SCROLLER Example

Shows how to use the TScroller class.

Project file

SCROLLER.IDE

Source files

SCROLLEX.CPP

SLIDER Example

Shows how to use the TSlider class.

Project file

SLIDER.IDE

Source files

SLIDERX.CPP

STATIC Example

Shows how to use the TStatic class.

Project file

STATIC.IDE

Source files

STATICX.CPP

VALIDATE Example

Shows how to use the TValidator class.

Project file

VALIDATE.IDE

Source files

VALIDATX.CPP

VBXCTL Example

Shows how to use the TVbxControl class.

Project file

VBXCTL.IDE

Header files

VBXCTL.H

Source files

VBXCTLX.CPP

CMDENABL Example

Shows how to use the TCommandEnabler class to enable and disable commands. Commands are either menu items, buttons on a control bar, or controls on a dialog box.

TCommandEnablers require a change in thinking about enabling and disabling commands. Rather than explicitly putting statements in your application to enable and disable commands, OWL will routinely ask your application if it should enable or disable the command. In addition to enabling and disabling commands, you can also use the command enabler to check a menu item or set its text.

Project file

CMDENABL.IDE

Header files

CMDENABL.H

Source files

CMDENAB1.CPP

CMDENAB2.CPP

CMDENAB3.CPP

CMDENAB4.CPP

UPDOWN Example

Shows how to use the TUpDown class.

Project file

UPDOWN.IDE

Source files

UPDOWNX.CPP

IMAGELST Example

Shows how to use an image list.

Project file

CELAPP.IDE

Source files

CELAPP.CPP

HOTKEY Example

Shows how to use the THotKey class.

Project file

HOTKEY.IDE

Header files

HOTKEY.H

Source files

HOTKEYX.CPP

TABCTRL Example

TTabControl encapsulates a Windows Tab Control. A Tab Control consists of one or more tab items, which are comparable to the dividers of a notebook, and a client area, which typically 'houses' a tab page. Each page is typically a window or dialog.

Project file

TABCTRL.IDE

Header files

TABCTRL.H

Source files

TABCTRLX.CPP

RICHEDIT Example

Shows how to use the TRichEdit class.

Project file

RICHEDIT.IDE

Header files

RICHEDAP.H

Source files

RICHEDAP.CPP

DOCKING Example

Shows how to create and use dockable control bars.

Project file

DOCKING.IDE

Header files

DOCKING.H

Source files

DOCKINGX.CPP

MRU Example

The MRU example shows how to use TRecentFiles, which encapsulates a most-recently-used file list.

TRecentFiles is designed to mix in with an TApplication. A mix-in class is a base class that is not commonly used for derivation. In the MRU example, TSampleApp is the derived class that has two base classes, TApplication and TRecentFiles. TApplication is the base class commonly derived from and TRecentFiles is the mix-in class. A mix-in is different from simple derivation because it allows you to decide whether you want to have a recently-used file list or not in your application. Mix-ins are derived from TEventHandler so that they can catch messages.

The class TRecentFiles works off of command enabling of the exit menu choice (which must have the id of either CM_FILEEXIT or CM_EXIT). If your application does not have a menu choice of those IDs, TRecentFiles will not work.

To add a choice to the menu, call TRecentFiles::SaveMenuChoice(). To know when the user clicked on one of the choices on the menu, TRecentFiles will send a registered message (MruFileMessage) to the main window. The WPARAM sent along with this message is the id you should pass to TRecentFiles::GetMenuText(), which gets the text of the menu choice. TSampleApp displays the selection in a message box.

Project file

MRU.IDE

Source files

MRU.CPP

SPLITTER Example

Shows how to use the TPaneSplitter class.

Project file

SPLITTER.IDE

Source files

SPLITTER.CPP

PROPSHT Example

Shows how to use property sheets (TPropertySheet) and property pages (TPropertyPage).

Project file

PROPSHT.IDE

Header files

PROPSHT.H

Source files

PROPSHTX.CPP

PICKLIST Example

The PICKLIST example shows how to use the TPickList class. Similar in functionality to the TInputDialog class, TPickList allows you to select an item from a list. Like TInputDialog, you'll need to #include <owl/picklist.rc> in your resource file.

Project file

PICKLIST.IDE

Source files

PICKLISX.CPP

SPLASH Example

The SPLASH example illustrates how to create a splash screen. There are several styles that can be used with TSplashWindow: None, ShrinkToFit, MakeGauge, MakeStatic, and CaptureMouse. They can be used in any combination. TSplashWindow's constructor also has a timeout parameter which signifies the number of milliseconds to display the splash window.

The ShrinkToFit refers to shrinking the splash window to fit around the dib. If this style is on, the width and height parameters to the constructor of TSplashWindow is ignored.

Project file

SPLASH.IDE

Source files

SPLASHX.CPP

TOOLTIP Example

Shows how to use the TTooltip class.

Project file

TOOLTIP.IDE

Header files

TOOLTIP.H

Source files

TOOLTIPX.CPP

ANIMCTL Example

Shows the basic functionality of the animation control. It shows how to store an AVI file as a resource and playing that AVI file using the TAnimateCtl class.

Project file

ANIMCTL.IDE

Header files

ANIMCTL.H

Source files

ANIMCTLX.CPP

DIBWIND Example

The DIBWIND example shows how to use the TDibWindow class.

Project file

DIBWIND.IDE

Source files

DIBWINDX.CPP

GADGETS Example

The GADGETS example illustrates how to use all of the available gadgets in OWL. A gadget is a user-interface element that acts like a window, but due to resource issues, it is not a window.

Sample Overview

The GADGETS example creates a decorated SDI main window. The example creates a control bar and inserts a button gadget, a separator gadget, a menu gadget, two mode gadgets, a text gadget and a control gadget wrapped around an edit control. The example creates a status bar with several mode gadgets, separator gadgets and a time gadget. Various styles of the gadgets are demonstrated.

The client window is the command target of the edit control gadget. Whenever the edit control sends a notification, the control gadget routes the notification to its command target, which is the client window.

Project file

GADGETS.IDE

Source files

GADGETS.CPP

METAFILE Example

Shows how to work with metafiles.

Source files

METAFILX.CPP

COLMNHDR Example

The ObjectsWindows classes TColumnHeader and THeaderItem can be used to create, alias and manipulate Header Controls. Header Controls are windows typically used to 'label' columns of data. The control contains one or more items, each label a column. Each Header Item can consists of a string, a bitmapped image and an associated application-defined 32-bit value.

Creating a Header Control

To create a Header Control, simply create a 'TColumnHeader' object within the constructor of the parent object specifying the 'parent' and the 'id' of the control.   For example,

TMyWindow::TMyWindow() 31580   // ...   TColumnHeader* hdrCtl = new TColumnHeader(this, ID_XX);   }

When constructed within the constructor of the parent object, it is not necessary to invoke the 'Create' method of the 'TColumnHeader' object. The 'AutoCreate' feature of ObjectWindows will ensure that the control is created once the parent object is created. (For more information on 'AutoCreation' see the 'xxxx' article //!BB). However, if you are constructing the TColumnHeader object after its parent has been created, you will also need to invoke it's 'Create' method.

NOTE: Although you can specify the screen coordinates of the control, these parameters are typically left out in favour of the 'Layout' capabilities of the 'TColumnHeader' object; more on this later.

Adding items to a Header Control

Once the control has been created (NOTE: It's important to distinguish between the construction of the C++ object and the creation of the actual underlying window - See the article 'Constructing/Destructing vs.   Creating/Destroying an ObjectWindows' for more information), you can add items using the 'Add' or 'Insert' methods of the TColumnHeader class.

The 'THeaderItem' class holds information about an item of a Header Control. To add/insert an item you must first construct a 'THeaderItem' instance. After initializing the 'THeaderItem' instance, you can invoke the 'Add' or 'Insert' method of the 'TColumnHeader' object. For example:

```
THeaderItem item("&Name of Employee");   hdrCtl->Add(item);
```

Responding to Header Control Notification Messages

The Header Control sends notification messages to its parent window whenever user manipulates the Header Control. For example, if the user clicks an item, the control sends a HDN_ITEMCLICK notification message.

ObjectWindows provides several macros which can be used in the definition of a Message Response Table allowing a member function to be invoked when particular notification messages are received by the parent. The following list the macros pertinent to the Header Control:

```
EV_HDN_BEGINTRACK(id, method)       // User starts dragging divider
  EV_HDN_DIVIDERDBLCLICK(id, method)  // User double clicked divider
  EV_HDN_ENDTRACK(id, method)         // User ends drag operation
  EV_HDN_ITEMCHANGED(id,method)       // Attribute of an item changed
  EV_HDN_ITEMCHANGING(id,method)      // Attribute about to change
  EV_HDN_ITEMCLICK(id, method)        // User clicked on item
EV_HDN_TRACK(id, method)            // User dragged a divider
```

Sizing and Positioning a Header Control

A Header Control is typically docked to the upper side of its parent's client area. The control provides an API which allows the control to specify a desired size and position within the boundary of a specified rectangle. The 'bool Layout(TRect& boundingRect, WINDOWPOS& winPos)' method can be used to retrieve the appropriate size and position values in a WINDOWPOS structure.

The overloaded 'bool Layout(uint swpFlags = 0)' method provides an higher abstraction of this API: the desired size of the control is retrieved specifying the client area of its parent as the bounding rectangle and the control is then repositioned accordingly. The 'swpFlags' are used when call 'SetWindowPos' to reposition the control.

Project file

COLMNHDR.IDE

Header files

COLMNHDR.H

Source files

COLMNHDX.CPP

UIFACE Example

Shows how to use the TUIFace class.

Source files

UIFACEX.CPP

UIFACE.CPP

UIBORDER Example

Shows how to use the TUIBorder class.

Project file

UIBORDER.IDE

Source files

UIBORDEX.CPP

DIBITMAP Example

Shows how to use the TDiBitmap class.

Project file

DIBITMAP.IDE

Source files

DIBITMAX.CPP

DRAGLIST Example

This sample illustrates how to use the TDragList class which provides the a draggable listbox. The sample creates a main window that has two children: a static control and the draggable listbox. The example derives a class TExampleDragList from TDragList to implement its features.

The static control displays information about the program as you are dragging items around.

The draggable listbox contains several text items. The first item cannot be dragged nor can it be dropped on. This is implemented by handling the virtual functions BeginDrag() and Dragging() in TExampleDragList. The second item in the listbox displays a copy cursor when it is dragged. Look in the Dragging() method to see how this is done. The other items in the listbox displays a regular move cursor when it is dragged.

DragList UI

To drag an item, click the left mouse button on the item and while holding the left mouse button, move the mouse to the destination. When you let go of the left mouse button, you drop the item. To cancel a drag, you can either click the right mouse button or press the Esc key while still holding the left mouse button.

General UI dictates that a regular drag moves the object. A ctrl-drag (i.e. holding the Ctrl key while dragging the mouse) copies the object. A shift-drag (i.e. holding the shift key while dragging the mouse) extends selection. Note that dragging and selecting items can be confusing to the user. Try not to mix both metaphors.

Source files

DRAGLISX.CPP