Zinc™ Interface Library™

Addendum

Version 1.01

Zinc Software Incorporated
Pleasant Grove, Utah

CONTENTS

NEW FEATURES OVERVIEW

UI_DOS_BGI_DISPLAY arguments

The UI_DOS_BGI_DISPLAY constructor now allows display modes other than the default. For a list of the available modes, refer to function 'initgraph' of the Turbo C++ Reference Guide.

The syntax of the UI_DOS_BGI_DISPLAY constructor is:

UI_DOS_BGI_DISPLAY(int *display*, int *mode*);

The arguments of the constructor are:

- *display* is an integer value that specifies the graphics driver to be used.
- *mode* is an integer value that specifies the graphics mode to be used.

For example, the following code would be used to construct a new UI_DOS_BGI_DISPLAY object that uses a CGA display with a CGAC2 mode:

```
#include <graphics.h>
main()
{
    UI_DOS_BGI_DISPLAY display(CGA, CGAC2);
    .
    .
    .
}
```

Using VROOMM with ZIL

The large memory model library has now been compiled with the *-Y* option so that programs can be linked with Borland's Turbo C++ VROOMM, allowing overlay support. For more information about using VROOMM, refer to the Turbo C++ Programmer's Guide.

New sample programs

Several new example programs have been added. The default directory for the sample programs is \ZINC\EXAMPLES. To make the new examples, enter:

make xcalc.exe—An example of a simple calculator.

make xclock.exe—An example of deriving a new device to create a clock.

make xgraphics.exe—An example of extended graphics using graph and pie charts.

make xwscroll.exe—An example using scroll bars.

Event Mapping

Extended documentation of Zinc Interface Library's event mapping is included in this addendum.

Scroll bar

Zinc Interface Library now supports text and matrix scrolling. Documentation for this new class is included in this addendum.

Generic Static Functions

A GENERIC function has been added to Zinc Interface Library that simplifies the creation of basic windows and system buttons. This new function is also explained in this addendum.

INSTALLATION

Installation of Zinc Interface Library requires DOS 2.1 or later (DOS 3.1 or later is recommended), 640K RAM and a hard disk drive. Before installing Zinc Interface Library, we recommend that you back-up your distribution disks.

Insert the first distribution disk into the desired drive, make it the current drive and invoke the installation program. For example, to install Zinc Interface Library from drive A, insert the first disk and type: **a:<Enter>**, then **install<Enter>**.

Pressing <Esc> at any time will cause the installation program to abort.

The install process is accomplished in five steps:

Confirmation of license agreement—If you wish to proceed and accept the agreement, select "yes." Otherwise, select "no" and the program will abort.

Selecting a drive—Select a hard disk drive for installation.

Selecting a subdirectory—Simply press <Enter> to select the default directory (\ZINC) or type in the desired directory and then press <Enter>.

Selecting portions to install—Selecting "yes" for any of the following options will install that portion of the library to your hard drive:

Demo	Utility Programs
Examples	Library Files
Include Files	Tutorials

Installation—The program now commences installing the selected material from the distribution disks to your hard drive. Periodically, a prompt for a new disk will appear. Remove the current disk from the drive, insert the disk requested and press any key to continue the installation.

At the end of this process, a message appears on your screen indicating that installation of Zinc Interface Library is now complete.

LIBRARY UPDATES

The following revisions have been made in Zinc Interface Library:

1. The mouse position now performs correctly when switching between graphics and text modes.

2. Moving among large numbers of elements in a matrix is now possible.

3. The text editor now allows sizing windows from very small to large.

4. Filling a string buffer to its maximum no longer causes problems.

5. The float and double number objects now allow fixed point calculations.

6. The UI_TIME_EXPORT now allows hundredths to default to null.

EVENT MAPPING

Event mapping within Zinc Interface Library consists of communication between at least two objects. One object sends a message, and eventually it is received by the object that can process the information. The receiving object attempts to perform the indicated action and then returns a message to the sender. The return message will fall into one of the four categories: 1) detection of an error; 2) unrecognized event; 3) absence of window object; or 4) confirmation of action completed. Since each object is independent from all others, this type of communication is necessary so that events will be processed efficiently.

Each event processed by Zinc Interface Library is contained in a UI_EVENT structure. The specific messages are contained in a *type* member variable of this structure. In addition to the *type* member variable, the following members are included in the UI_EVENT structure:

```
struct UI_EVENT
{
    int type;
    USHORT rawCode;
    union
    {
        UI_KEY key;
        UI_REGION region;
        UI_POSITION position;
        UI_SCROLL_INFORMATION scroll;
        void *data;
    }
};

struct UI_KEY
{
    UCHAR shiftState;
    UCHAR value;
};

struct UI_REGION
{
    int left;
    int top;
    int right;
    int bottom;
};

struct UI_POSITION
{
    int column;
    int line;
};
```

```
struct UI_SCROLL_INFORMATION
{
    int current;
    int showing;
    int maximum;
    int delta;
};
```

The logical and system event mapping of Zinc Interface Library handles the following messages (contained in the *type* member variable of the UI_EVENT structure):

S_ALT_KEY—Indicates that the keyboard <Alt> key was pressed and then released without another key being pressed. This message is only sent by the keyboard device, and it causes the window's system menu (if any) to become current.

S_CANCEL—Can be processed by the programmer to cancel operations done to the current window. Unless this event is implemented by the programmer, it is ignored by the window manager, since complete backup copies of a window's field data are not available within the library itself.

S_CHANGE—Is passed by a window object to the window manager to tell it that the size of its parent window (the window to which it is attached) should be automatically changed without any user input. The new size of the window object is determined by the UI_REGION portion of the UI_EVENT structure.

S_CHECK_HOT_KEY—Is a message sent by a parent window to a child object (an object attached to the window), telling it to check the passed event with the its list of hot keys to see if any match. Typically this message is used by a window with a menu field where hot keys may be used to change the position of the highlight within the window. In other words, if a hot key is pressed by an end user, the message is sent through the window manager to the current window. The parent window passes the message to each of its window objects until a match is confirmed by one of the objects or else is disconfirmed by all objects. If a match is confirmed, the object containing the hot key match becomes current and selected.

S_CONTINUE—This message is sent by the programmer to the event manager, allowing it time to poll its devices (e.g., keyboard and cursor). The message is sent to the current object but does not have any effect on program execution other than to allow the system time to poll the devices. For example, this event is necessary if a continuous clock device has been implemented, because it ensures that the clock will change every second even if another time-consuming event is in progress.

S_CREATE—Tells each window object to initialize its internal information, such as its size and position within the parent window. The S_CREATE message is always succeeded by an S_CURRENT, S_DISPLAY_ACTIVE or S_DISPLAY_INACTIVE message. This message is sent to all of the window objects associated with a window whenever the window is attached to the window manager.

S_CURRENT—Is a message, sent through the window manager, which tells the receiving window object that it is the current (or highlighted) window object on the screen. This message is sent instead of the S_DISPLAY_ACTIVE message, which applies to all other objects within the window. The S_CURRENT message uses the UI_REGION portion of the UI_EVENT structure to indicate the coordinates of any region that overlaps a part of the previously current window object. This region will need to be refreshed, or repainted, to the foreground of the display. If no region is overlapping, the object will simply be shown in a current mode, without any repainting. Once the S_CURRENT message is sent, the window object will receive all relevant event information passed through the window manager, since event messages are most often intended for the current object.

S_DELETE—Tells each window object to uninitialize its internal information. This message is sent by the window manager whenever a window is removed from the screen display (such as when the close option is selected by the end-user). This message can also be sent directly to the window manager, telling it to remove the current window from its list of windows and to destroy it. As a result, the window is removed from the screen. If a temporary window (a window with the WOAF-_TEMPORARY flag set, such as a pull-down menu) is present on the screen, it will remove it and the next window in the window manager's list

of windows. If the current window is locked (i.e., it cannot be removed from the window manager) the S_DELETE message makes the next object on the screen current. If the WOAF_NO_DESTROY flag is set on any window, the S_DELETE message will cause that window to be subtracted from the window manager, which will cause it to disappear from the screen as well, but it will not be destroyed.

S_DELETE_LEVEL—Is the same as S_DELETE, except that it always removes only one window from the screen. Thus, if the top window is a temporary window (as determined by the WOAF_TEMPORARY flag), it is the only window removed from the screen. For example, if an end user presses <Esc> (the default key associated with S_DELETE_LEVEL) while a pull-down menu is the current window, only the pull-down menu will be removed. Any other windows will remain on the screen. In all other instances, the current window is removed from the screen.

S_DISPLAY_ACTIVE—Is sent through the parent window to a window object, telling the object to re-display itself according to an active state. In other words, one of the other objects within the parent window has become current, so the rest of the objects must be re-displayed according to an active state. This message is passed with the affected region (contained in the UI_REGION portion of the UI_EVENT structure). The object only needs to re-display its screen information when the region passed by the event overlaps the region of the object.

S_DISPLAY_INACTIVE—Is sent through a parent window to a window object, telling the object to redisplay itself according to an inactive state. In other words, the parent window is no longer active (meaning that none of its objects are current), and all objects within it must be re-displayed as inactive. This message is passed with the affected region (contained in the UI_REGION portion of the UI_EVENT structure). The object only needs to re-display its screen information when the region passed by the event overlaps the region of the object.

S_ERROR—Is returned by the object to the window manager, indicating that an error has been identified by the object and that the event should not be processed. In the meantime, the object also calls the error system, which performs in one of the following two ways:

1—If the default error system is in effect, calling the error system will cause a "beep."

2—If the window error system has been initialized by the programmer, a modal error window will appear, which remains current until the end user chooses an option—either continue, unanswered or invalid.

For example, if an end-user enters an out-of-range date in a date field and then presses <Enter>, the object detects an error, tells the window manager to halt the process, and calls the error system. When the end-user responds to the error warning appropriately, the error system sends the S_ERROR_RESPONSE to the object, withdraws, and the next window object is made current.

S_ERROR_RESPONSE—Is sent by the error system to the object where the error occurred when the end-user responds to an error message. This in turn causes the modal error window or the default error system to withdraw.

S_MAXIMIZE—Is sent through the window manager to the parent window, telling it to either enlarge the current window on the screen to its maximum size (normally the size of the screen), or to restore the window to its normal state if it is already maximized.

S_MINIMIZE—Is sent through the window manager to the parent window, telling it to either reduce the current window on the screen to its minimum size, or to restore the window to its normal state if it is already minimized.

S_MOVE—Is sent from the window manager to the window object, telling it that the parent window has been moved. The delta position of the object is contained in the *position* field of UI_EVENT, which is passed within this message. For example, an *event.position.line* of -10 and an *event.position.column* of 15 says that the window object coordinates need to be moved 10 lines up and 15 columns to the right.

S_NO_OBJECT—Is sent back to the programmer from the window manager, indicating that no window object is attached to it. An event was

passed that was intended for a particular window object, but since no object is attached to the window manager, no processing can occur.

S_NON_CURRENT—Tells the receiving window object that it is no longer the current window object. The only exception to this is when the receiving object sends back an S_ERROR message (e.g., a date field with an out-of-range date). In this case, the current window object does not change its status until the error is resolved.

S_REDISPLAY—Is handled by the window manager. It tells the system to re-display the background and all windows attached to the screen. This message is converted to S_CURRENT, S_DISPLAY_ACTIVE and S_DISPLAY_INACTIVE messages by the window manager and is sent as such to each window that is attached to the window manager.

S_SCROLL_VERTICAL—Tells the receiving window object to scroll its information the total number of lines specified by *event.scroll.delta*. For example, a value of -5 tells the receiving object to scroll the information up five lines.

S_SIZE—Is passed by window objects to the window manager in order to initiate a size operation. In this case, the window manager allows the end-user to size the window according the size operations permitted by the application. The UI_REGION portion of UI_EVENT structure is used to indicate which sides of the window can be modified. The available selections (M_RIGHT_CHANGE, M_TOP_CHANGE, M_BOTTOM-_CHANGE, M_LEFT_CHANGE) are OR'ed together to give the allowed size operations. Once a window is sized, the S_SIZE message is passed on to the affected window to indicate the new size of the window. The new window size is once again contained in the UI_REGION portion of the UI_EVENT structure.

S_UNKNOWN—The event (keyboard, mouse or system) was not recognized by the receiving window object. If this message is passed back, no action was taken by the current window object.

L_BEGIN_SELECT—Begins the selection process of a window or window object. (This message is normally sent by the mouse driver.) For

example, if the end-user clicks down on the left mouse button, the selection of the object is initiated. When the mouse button is released (L_END_SELECT), the selection will be completed.

L_CONTEXT_HELP—Requests help about a particular window. The current window calls the window manager, which in turn displays context sensitive help in its information window.

L_CONTINUE_SELECT—Is a drag operation sent by the mouse driver, indicating that L_BEGIN_SELECT was already sent and the mouse is presently being dragged as part of the selection process.

L_END_SELECT—Indicates that the selection process, initiated with the L_BEGIN_SELECT message, is complete. For example, the end-user has pressed and released the left mouse button.

L_EXIT—Tells the programmer that all program flow should discontinue between the event manager and the window manager. The window manager treats this message as a no-op; that is, no windows are removed from the window manager until its destructor is called.

L_GENERAL_HELP—Asks for general help associated with the application. This message is processed by the window manager, which in turn calls the global help system.

L_SELECT—Selects the current object or completes a selection process. For example, if you are positioned on a menu item and press <Enter> the message is converted to an L_SELECT logical event, and the item is selected and any associated action procedure is called.

L_WINDOW_MOVE—Is the same as S_MOVE, except that the event is initiated by the end-user rather than by another window object. For example, if the end-user presses <Alt F7>, which is the default key associated with this operation, the window manager will pass the message down to each window object so that it knows its new coordinates.

L_WINDOW_NEXT—Makes the next window object the current window object on the screen.

L_WINDOW_RESTORE—Restores the window object to its original size.

L_WINDOW_SIZE—Is the same as S_SIZE, except that the event is generated by the end-user rather than by another window object. For example, if the end-user presses <Alt F8>, which is the default key associated with this operation, the message is passed to the window for resizing.

L_VIEW—Is an interpreted event which indicates that no mouse buttons are currently pressed but the mouse is being moved across the screen.

All of the following are movement for window objects which are attached to a window:

L_FIELD_DOWN—If the field occupies a single line on the screen, or if the cursor is positioned on the bottom line of a multi-line field, L_FIELD_DOWN moves from the current window field to the window field immediately *below* the current field. The left or right boundary of the field must be aligned vertically with the left or right boundary of the current field. If the field is a multi-line field and the cursor is not positioned on the bottom line, L_FIELD_DOWN moves the cursor *down* one line on the display.

L_FIELD_FIRST—Moves to the first field of the current window.

L_FIELD_LAST—Moves to the last field of the current window.

L_FIELD_LEFT—Moves the cursor to the beginning of the field or object to the left of the current field. The border of the left field must be aligned horizontally with the border of the current field.

L_FIELD_NEXT—Moves from the current window field to the *next* selectable window field (i.e., the next window field that is attached to the window manager). If the last window field is currently selected, L_FIELD_NEXT cycles to the *first* selectable window field.

L_FIELD_PREVIOUS—Moves from the current window field to the *previous* selectable window field (i.e., the window field attached

immediately before in the window manager). If the first window field is currently selected, L_FIELD_PREVIOUS cycles to the *last* selectable window field.

L_FIELD_RIGHT—Moves the cursor to the beginning of the field or object to the right of the current field. The border of the right field must be aligned horizontally with the border of the current field.

L_FIELD_UP—If the field occupies a single line on the screen, or if the cursor is positioned on the top line of a multi-line field, L_FIELD_UP moves from the current window field to the window field immediately *above* the current field. The left or right boundary of the field must be aligned vertically with the left or right boundary of the current field. If the field is a multi-line field and the cursor is not positioned on the top line, L_FIELD_UP moves the cursor *up* one line on the display.

All of the following are movement for menu items:

L_ITEM_DOWN—Moves to the menu item immediately *below* the current menu item.

L_ITEM_FIRST—Moves to the first item of the current menu, or, if the first item is already current, it moves to the first item of the previous selectable menu.

L_ITEM_LAST—Moves to the last item of the current menu, or, if the last item is already current, it moves to the last item of the next selectable menu.

L_ITEM_LEFT—If the current menu has more than one column of items, L_ITEM_LEFT moves the cursor (highlight) to the item immediately to the left of the current item.

L_ITEM_NEXT—Moves from the current menu item to the next selectable menu item.

L_ITEM_PREVIOUS—Moves from the current menu item to the previous selectable menu item.

L_ITEM_RIGHT—If the current menu has more than one column of items, L_ITEM_RIGHT moves the cursor (highlight) to the item immediately to the right of the current item.

L_ITEM_UP —Moves to the menu item immediately above the current menu item.

The remaining logical events are processed by the edit objects (UIW_STRING, UIW_TEXT, UIW_NUMBER, UIW_DATE, UIW_TIME).

L_BEGIN_MARK—Determines the beginning position in marking a region.

L_CONTINUE_MARK—Indicates that the marking procedure (initiated by L_BEGIN_MARK) is in progress and that the marked area is being enlarged or reduced.

L_COPY_MARK—Copies the marked region and places it in the global paste buffer.

L_CUT—Removes the marked region and places it in the global paste buffer.

L_CUT_PASTE—If a region is marked, this message cuts the marked region and places it in the global paste buffer. If a region is not marked, this message pastes the contents of the global paste buffer at the current cursor position.

L_DELETE—Deletes the character at the cursor position. The cursor position remains the same after the operation.

L_DELETE_EOL—Deletes from the cursor position to the end of the line.

L_DELETE_WORD—Deletes the word at the current cursor position.

L_END_MARK—Completes the mark operation (initiated by L_BEGIN_MARK) of a specific region within the field.

L_INSERT_TOGGLE—Toggles between *insert* and *overstrike* edit modes in a field.

L_MARK—Marks *all* of the information contained in the field. This message is generally only interpreted by one line edit objects.

L_MOVE_BOL—Moves the cursor to the beginning of the current line within the current object.

L_MOVE_BOTTOM—Moves the cursor to the bottom of the current field.

L_MOVE_DOWN—If the field occupies a single line of the screen, or if the cursor is positioned on the bottom line of a multi-line field, L_MOVE_DOWN moves from the current window field to the window field immediately below the current field. The left or right edge of the field above must be aligned vertically with the boundary of the current field. If the field is a multi-line field and the cursor is not positioned on the bottom line, L_MOVE_DOWN moves the cursor down one line on the display.

L_MOVE_EOL—Moves the cursor to the end of the current line within the current object.

L_MOVE_LEFT—Moves the cursor to the previous character.

L_MOVE_PAGE_DOWN—If the field occupies a single line on the screen, or if the cursor is positioned on the bottom line of a multi-line field, L_MOVE_PAGE_DOWN moves the cursor from the current window field to the last window field. If the field is a multi-line field and the cursor is not positioned on the bottom line, L_MOVE_PAGE_DOWN moves the cursor down one page in the current field.

L_MOVE_PAGE_UP—If the field occupies a single line on the screen, or if the cursor is positioned on the top line of a multi-line field, L_MOVE_PAGE_UP moves the cursor from the current window field to the first window field. If the field is a multi-line field and the cursor is not positioned on the top line, L_MOVE_PAGE_UP moves the cursor up one page in the current field.

L_MOVE_RIGHT—Moves the cursor to the next character.

L_MOVE_TOP—Moves the cursor to the top of the current field.

L_MOVE_UP—If the field occupies a single line of the screen, or if the cursor is positioned on the top line of a multi-line field, L_MOVE_UP moves form the current window field to the window field immediately above the current field. The left or right edge of the field above must be aligned vertically with the boundary of the current field. If the field is a multi-line field and the cursor is not positioned on the top line, L_MOVE_UP moves the cursor up one line on the display.

L_PASTE—Copies the contents of the global paste buffer (placed there by the L_CUT or L_COPY_MARK procedures) into the current field. The copy will only occur if the data matches the type of information that the field can receive.

L_REDO—Restores, in the current field, the most recent changes executed by the L_UNDO procedure.

L_UNDO—Undoes the last operation of the field.

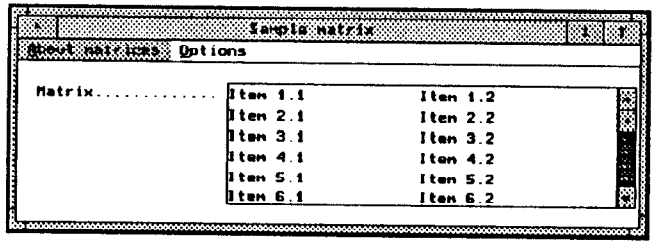
L_WORD_TAB_LEFT—Moves the cursor to the beginning of the previous word.

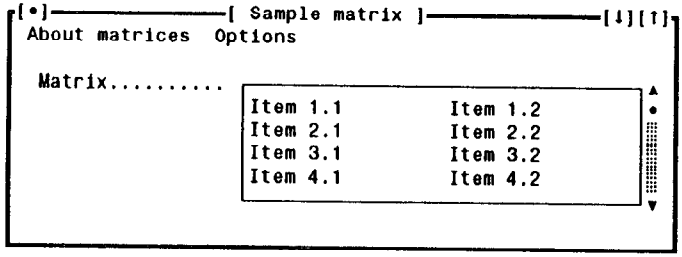
L_WORD_TAB_RIGHT—Moves the cursor to the beginning of the next word.

UIW_SCROLL_BAR

Overview

The UIW_SCROLL_BAR class is used to change displayed information in an associated text or matrix field so that additional data which is hidden above or below the displayed portion can be seen. The figures below show graphic and textual implementations of a window with a UIW_SCROLL-_BAR class object:





Mouse interaction with the scroll bar produces the following effects:

Clicking on the up arrow (▲) causes the data to scroll up one line.

Clicking on the down arrow (▼) causes the data to scroll down one line.

Clicking on and dragging the slider box (identified by a '•' character) causes the data to scroll to a position

proportional to the new slider box position. (The position is changed when the mouse button is released.)

Clicking on empty space above the slider box causes the data to scroll one page up. Likewise, clicking on empty space below the box causes the data to scroll one page down.

When the scroll bar detects input from the mouse, it sends the **L_SCROLL_VERTICAL** message to the receiving window object (i.e., the text or matrix field with which it interacts). The *event.scroll.delta* portion of the UI_EVENT structure (which is used in communicating the **L_SCROLL_VERTICAL** message) specifies the total number of lines that the information should scroll. For example, a value of -5 tells the receiving object to scroll the information **up** five lines.

Declaration The public and inherited members of the UIW-_SCROLL_BAR class (declared in **UI_WIN.HPP**) are:

```
class UIW_SCROLL_BAR : public UIW_WINDOW
{
public:
USHORT sbFlags;

    UIW_SCROLL_BAR(int left, int top, int width,
        int height, USHORT sbFlags, USHORT woFlags;
    virtual ~UIW_SCROLL_BAR(void) {}

    virtual int Event(const UI_EVENT &event);
};


class UIW_WINDOW : public UI_WINDOW_OBJECT
{
public:
    UIW_WINDOW(int left, int top, int width,
        int height, USHORT woFlags,
        USHORT woAdvancedFlags,
        int helpContext = NO_HELP_CONTEXT);
    virtual ~UIW_WINDOW(void);

    void Add(UI_WINDOW_OBJECT *object);
    UI_WINDOW_OBJECT *First(void);
    UI_WINDOW_OBJECT *Last(void)
    void Subtract(UI_WINDOW_OBJECT *object);
```

```
        UIW_WINDOW &operator + (void *object);
        UIW_WINDOW &operator - (void *object);
    };

class UI_WINDOW_OBJECT : public UI_ELEMENT
{
public:
    static UI_EVENT_MAP *eventMapTable;
    static int defaultDepth;

    USHORT woFlags;
    USHORT woStatus;
    UI_REGION true;
    UI_WINDOW_OBJECT *parent;
    UI_DISPLAY *display
    UI_EVENT_MANAGER *eventManager;
    UI_WINDOW_MANAGER *windowManager;
    UI_PALETTE_MAP *paletteMapTable;

    UI_WINDOW_OBJECT *Next(void);
    UI_WINDOW_OBJECT *Previous(void);
};

class UI_ELEMENT
{
public:
    UI_ELEMENT *previous;
    UI_ELEMENT *next;

    UI_ELEMENT(void);
    virtual ~UI_ELEMENT(void);
};
```

See also The example file **XWSCROLL.CPP**, which gives a complete example of the UIW_SCROLL_BAR class.

UIW_SCROLL_BAR::UIW_SCROLL_BAR

Syntax

```
#include <ui_win.hpp>

UIW_SCROLL_BAR::UIW_SCROLL_BAR(int left, int top,
int width, int height, USHORT sbFlags, USHORT woFlags);
```

Remarks This constructor returns a pointer to a new UIW_SCROLL_BAR class object.

- *width*$_{in}$ is the width of the scroll bar. This value is determined automatically by the UIW_SCROLL_BAR class object when the SBF_VERTICAL flag is set. A value, however, must be entered, even though it will be ignored.

- *height*$_{in}$ is the height of the scroll bar. This value will be ignored if the WOF_NON_FIELD_REGION flag is set.

- *sbFlags*$_{in}$ gives information on how to display the scroll bar. The following flag (declared in **UI_WIN.HPP**) controls the general presentation of a UIW-_SCROLL_BAR class object:

  SBF_VERTICAL—Displays a vertical scroll bar.

- *woFlags*$_{in}$ are flags (common to all window objects) that determine the general presentation of the scroll bar object. The following flags (declared in **UI_WIN.HPP**) control the general presentation of a UIW-_SCROLL_BAR class object:

  WOF_BORDER—Draws a single line border around the scroll bar object. This flag should only be used when the WOF_NON_FIELD_REGION flag is set.

  WOF_NO_FLAGS—Does not associate any special flags with the scroll bar object. This flag should not be used in conjunction with any other WOF flag.

  WOF_NON_FIELD_REGION—The scroll bar object is not a form field. If this flag is set the scroll bar object will occupy all of the remaining space of its parent window. Note: If the text or matrix field to which the scroll bar is attached has a WOF_NON-_FIELD_REGION flag set, the scroll bar must also set this flag. Likewise, if the text or matrix field does

not have the WOF_NON_FIELD_REGION set, the scroll bar cannot have it set either.

To ensure that the scroll bar is drawn correctly, it must be created immediately *before* the object it affects. The following example shows the correct and incorrect order of scroll bar creation:

```
1)  // CORRECT construction order.
    UIW_WINDOW *menuWindow = new UIW_WINDOW(5, 5, 50, 12,
            WOF_NO_FLAGS, WOAF_NO_FLAGS);
    *menuWindow
        + new UIW_BORDER
        + new UIW_MAXIMIZE_BUTTON
        + new UIW_MINIMIZE_BUTTON
        + new UIW_SYSTEM_BUTTON
        + new UIW_TITLE("Sample Menu")
        + new UIW_SCROLL_BAR(14, 1, 1, 5, SBF_VERTICAL,
            WOF_BORDER)
        + new &(*new UIW_MATRIX(2, 1, 12, 5, 10,
            12, 1, 0, MXF_SELECT_ONE, WOF_BORDER,
            WOAF_NO_FLAGS)
            + new UIW_POP_UP_ITEM(0, 0, 12,
                " Option 1", MNIF_NO_FLAGS,
                BTF_CHECK_MARK, WOF_NO_FLAGS, 0)
            + new UIW_POP_UP_ITEM(0, 1, 12,
                " Option 2", MNIF_NO_FLAGS,
                BTF_CHECK_MARK, WOF_NO_FLAGS, 0)
              .
              .
              .

2)  // INCORRECT construction order.
    UIW_WINDOW *menuWindow = new UIW_WINDOW(5, 5, 50, 12,
        WOF_NO_FLAGS, WOAF_NO_FLAGS);
    *menuWindow
        + new UIW_BORDER
        + new UIW_MAXIMIZE_BUTTON
        + new UIW_MINIMIZE_BUTTON
        + new UIW_SYSTEM_BUTTON
        + new UIW_SCROLL_BAR(14, 1, 1, 5, SBF_VERTICAL,
            WOF_BORDER)
        + new UIW_TITLE("Sample Menu")
        + new &(*new UIW_MATRIX(2, 1, 12, 5, 10,
            12, 1, 0, MXF_SELECT_ONE, WOF_BORDER,
            WOAF_NO_FLAGS)
            + new UIW_POP_UP_ITEM(0, 0, 12,
                " Option 1", MNIF_NO_FLAGS,
                BTF_CHECK_MARK, WOF_NO_FLAGS, 0)
            + new UIW_POP_UP_ITEM(0, 1, 12,
                " Option 2", MNIF_NO_FLAGS,
                BTF_CHECK_MARK, WOF_NO_FLAGS, 0)
              .
              .
              .
```

NOTE: If the scroll bar is added to a parent window, it will automatically be destroyed when the parent window is destroyed.

Example

```
#include <ui_win.hpp>

ExampleFunction1()
{
    UIW_SCROLL_BAR *scrollBar =
        new UIW_SCROLL_BAR(14, 1, 1, 5,
            SBF_VERTICAL, WOF_BORDER);

    // Add the window objects to the window.
    *menuWindow
        + new UIW_BORDER
        + new UIW_MAXIMIZE_BUTTON
        + new UIW_MINIMIZE_BUTTON
        + new UIW_SYSTEM_BUTTON
        + new UIW_TITLE("Sample Menu",
            WOF_JUSTIFY_CENTER)
        + scrollBar
        + &(*new UIW_MATRIX(2, 1, 12, 5, 10,
            12, 1, 0, MXF_SELECT_ONE,
            WOF_BORDER, WOAF_NO_FLAGS)
            + new UIW_POP_UP_ITEM(0, 0, 12,
                " Option 1", MNIF_NO_FLAGS,
                BTF_CHECK_MARK,
                WOF_NO_FLAGS, 0)
            + new UIW_POP_UP_ITEM(0, 1, 12,
                " Option 2", MNIF_NO_FLAGS,
                BTF_CHECK_MARK,
                WOF_NO_FLAGS, 0)
        .
        .
        .

}
```

UIW_SCROLL_BAR::~UIW_SCROLL_BAR

Syntax

```
#include <ui_win.hpp>

virtual UIW_SCROLL_BAR::~UIW_SCROLL_BAR(void);
```

Remarks

This virtual destructor destroys the class information associated with the UIW_SCROLL_BAR object. Care should be taken to only destroy scroll bar objects that are not attached to a parent window.

Example

```
#include <ui_win.hpp>

ExampleFunction1()
{
    UIW_SCROLL_BAR *scrollBar =
        new UIW_SCROLL_BAR(14, 1, 1, 5,
            SBF_VERTICAL, WOF_BORDER);

    // Add the window objects to the window.
    *menuWindow
        + new UIW_BORDER
        + new UIW_MAXIMIZE_BUTTON
        + new UIW_MINIMIZE_BUTTON
        + new UIW_SYSTEM_BUTTON
        + new UIW_TITLE("Sample Menu",
            WOF_JUSTIFY_CENTER)
        + scrollBar
        + &(*new UIW_MATRIX(2, 1, 12, 5, 10,
            12, 1, 0, MXF_SELECT_ONE,
            WOF_BORDER, WOAF_NO_FLAGS)
            + new UIW_POP_UP_ITEM(0, 0, 12,
                " Option 1", MNIF_NO_FLAGS,
                BTF_CHECK_MARK,
                WOF_NO_FLAGS, 0)
            + new UIW_POP_UP_ITEM(0, 1, 12,
                " Option 2", MNIF_NO_FLAGS,
                BTF_CHECK_MARK,
                WOF_NO_FLAGS, 0)
        .
        .
        .

    // Remove the scroll bar.
    *menuWindow - scrollBar;
    delete scrollBar;
}
```

GENERIC STATIC FUNCTIONS

To simplify the code associated with windows, the concept of GENERIC static functions has been added to Zinc Interface Library. Currently, two high level objects have a static member function called GENERIC, namely UIW_WINDOW and UIW_SYSTEM BUTTON. The GENERIC function creates the object, automatically including several objects that are commonly attached to it. Besides simplifying the code, this new function saves time and memory.

The GENERIC function returns a pointer to the newly created object. The new operator is not needed, since the GENERIC function uses it internally.

The UIW_WINDOW object's creation using the UIW_WINDOW::GENERIC function includes a border, a maximize button, a minimize button, a system button, and a title. For example, the original code for creating a window that contains these window objects is:

```
// Create the hello world window.
UIW_WINDOW *window =
    new UIW_WINDOW(5, 5, 40, 6, WOF_NO_FLAGS, WOAF_NO_FLAGS,
        HELP_HELLO_WORLD);

// Add the window objects to the window.
*window
    + new UIW_BORDER
    + new UIW_MAXIMIZE_BUTTON
    + new UIW_MINIMIZE_BUTTON
    + new UIW_SYSTEM_BUTTON
    + new UIW_TITLE("Hello World Window", WOF_JUSTIFY_CENTER)
    + new UIW_TEXT("Hello, World!", 256, TXF_NO_FLAGS,
        WOF_NON_FIELD_REGION);
```

Using the UIW_WINDOW::GENERIC function, the above code can be replaced with:

```
// Create the standard Hello World! window.
UIW_WINDOW *window = UIW_WINDOW::GENERIC(2, 2, 40, 6,
    WOF_NO_FLAGS,   WOAF_NO_FLAGS, HELP_HELLO_WORLD,
    "Hello World Window");

// Add the window objects to the window.
*window
    + new UIW_TEXT("Hello, World!", 256, TXF_NO_FLAGS,
        WOF_NON_FIELD_REGION);
```

Creating a system button with the UIW_SYSTEM_BUTTON::GENERIC function, the button is first created, and then the following menu option objects are added: Restore, Move, Size, Minimize, Maximize, and Close. For example, the original code for creating such a system button is:

```
*window
    + &(*new UIW_SYSTEM_BUTTON
        + new UIW_POP_UP_ITEM("~Restore", MNIF_RESTORE,
            BTF_NO_TOGGLE, WOF_NO_FLAGS, 0)
        + new UIW_POP_UP_ITEM("~Move", MNIF_MOVE, BTF_NO_TOGGLE,
            WOF_NO_FLAGS, 0)
        + new UIW_POP_UP_ITEM("~Size", MNIF_SIZE, BTF_NO_TOGGLE,
            WOF_NO_FLAGS, 0)
        + new UIW_POP_UP_ITEM("Mi~nimize",
            MNIF_MINIMIZE, BTF_NO_TOGGLE, WOF_NO_FLAGS, 0)
        + new UIW_POP_UP_ITEM("Ma~ximize",
            MNIF_MAXIMIZE, BTF_NO_TOGGLE, WOF_NO_FLAGS, 0)
        + new UIW_POP_UP_ITEM
        + new UIW_POP_UP_ITEM("~Close", MNIF_CLOSE, BTF_NO_TOGGLE,
            WOF_NO_FLAGS, 0));
```

Using the UIW_SYSTEM_BUTTON::GENERIC function, the above code can be replaced with:

```
*window  + UIW_SYSTEM_BUTTON::GENERIC();
```

GNU Free Documentation License
Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input

to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy

a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains

nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit

corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME.
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.